Precomputing Oblivious Transfer

Donald Beaver

317 Pond Lab, Penn State University, University Park, PA 16802, (814) 863-0147;
beaver@cse.psu.edu.

Abstract. Alice and Bob are too untrusting of computer scientists to let their privacy depend on unproven assumptions such as the existence of one-way functions. Firm believers in Schrödinger and Heisenberg, they might accept a quantum OT device, but IBM's prototype is not yet portable. Instead, as part of their prenuptial agreement, they decide to visit IBM and perform some OT's in advance, so that any later divorces, coin-flipping or other important interactions can be done more conveniently, without needing expensive third parties.

Unfortunately, OT can't be done in advance in a direct way, because even though Bob might not know what bit Alice will later send (even if she first sends a random bit and later corrects it, for example), he would already know which bit or bits he will receive. We address the problem of *precomputing* oblivious transfer and show that OT can be precomputed at a cost of $\Theta(k)$ prior transfers (a tight bound). In contrast, we show that variants of OT, such as one-out-of-two OT, can be precomputed using only one prior transfer. Finally, we show that all variants can be reduced to a single precomputed one-out-of-two oblivious transfer.

1 Introduction

Oblivious transfer [Rab81], a process by which Alice sends Bob a bit over a noisy channel without knowing whether it arrives, is a fundamental and ubiquitous tool for cryptographic protocols. It comes in several varieties, but one thing is common to all implementations: they require intensive online computation relying on unproven intractability assumptions [Rab81, BM89, HL90, Boe91, Bea92], or they rely on not-yet-available hardware whose reliability is based on well-demonstrated physical facts, such as Heisenberg's Uncertainty Principle [BC90, BBCS91].

In this paper, we reduce online oblivious transfer to precomputed oblivious transfer. These reductions provide two immediate benefits. First, online computations such as generating large Blum integers or computing modular square roots can be performed in advance, using machine cycles and network bandwidth at off-peak times. (Note that we are distinctly *not* referring to *preprocessing*, such as computing a bunch of moduli and storing them for later use, but rather to *executing* the transfers already, by whatever means available, and storing the single-bit *results* to build later transfers when the desired bits are known.) Second, unproven assumptions can be avoided entirely by visiting Billes Grassard,[1]

[0] Supported in part by NSF grant CCR-9210954.

[1] Names have been changed to protect the innocent.

D. Coppersmith (Ed.): Advances in Cryptology - CRYPTO '95, LNCS 963, pp. 97-109, 1995.

	Precomputed		
$\leq$	OT	$\frac{1}{2}$OT	$\binom{2}{1}$OT
OT	$\Theta(k)$	1	1
$\frac{1}{2}$OT	$\Theta(k)$	1	1
$\binom{2}{1}$OT	$\Theta(k)$	1	1

Fig. 1. Tight costs of implementing one OT variant using another precomputed variant, to achieve $1-O(2^{-k})$ "security." Cost is measured as the number of invocations of the precomputed primitive.

asking for a couple of hours on the quantum OT machine [BBCS91], and transferring a set of bits for later use.

For subtle reasons, however, it is not at all immediate that OT can be precomputed. If Alice wishes to reserve her choice of a given bit b to send until tomorrow, then for whatever bits she sends today, Bob will already know which bits have arrived. For example, if she goes ahead and obliviously transfers a random bit r today, with the intent of sending $b \oplus r$ on a direct channel tomorrow, then even though Bob can't learn b until tomorrow, he does get some unfair information: he knows *today* whether he will get the result tomorrow or not!

We show that it is indeed possible to precompute the original OT introduced by Rabin [Rab81]. To achieve a $1-O(2^{-k})$ level of information-theoretic security, our protocols use $\Theta(k)$ prior transfers. The parameter k arises regardless of whether the precomputed OT's are implemented through cryptographic means or via a quantum channel. Our bound is tight: we also show that $\Omega(k)$ prior transfers are necessary.

We also consider the following variants:

- OT, or "standard" OT, permits Alice to send a bit b to Bob. Bob receives b with 50-50 probability and knows whether he received the bit. Alice does not know whether Bob received the bit.
- $\frac{1}{2}$OT, or "one-out-of-two" OT, permits Alice to send one of two bits, b_0 or b_1, to Bob [EGL82]. Bob receives either b_0 or b_1, with equal probability, and he knows which one he received. Alice does not know which bit Bob received.
- $\binom{2}{1}$OT, or "chosen one-out-of-two" OT, is similar to $\frac{1}{2}$OT, except that Bob chooses an index c and receives b_c. Alice does not learn c.

In contrast to OT, $\frac{1}{2}$OT and $\binom{2}{1}$OT can be precomputed at no additional cost, *i.e.* requiring only one prior transfer per future transfer. Moreover, any of the three variants can be reduced to a single precomputed $\frac{1}{2}$OT (or $\binom{2}{1}$OT), with no chance of error.

Figure 1 illustrates our results, demonstrating the costs of implementing any of the three standard variants using a primitive that is only temporarily available. Clearly, the optimal number of invocations made by an *online* reduction can be no larger than that by a precomputed reduction (else the precomputation would be done online!). Except for the precomputed self-reduction for OT, the costs of precomputation are no more than those of *online* reductions (*cf.*

[BCR86a, BCR86b, Cré87]; as part of our analysis, we also introduce simple online reductions between $\binom{2}{1}$OT and $\frac{1}{2}$OT).

These results shed some light on the nature of the three variants. Intuitively, OT requires us to maintain a degree of flexibility so that the fate of a future bit – *ie.* whether it will arrive or not – remains undetermined. Unfortunately, the fact of whether a bit arrived or not cannot be reversed later on. In contrast, the as-yet undetermined choices made in $\frac{1}{2}$OT and $\binom{2}{1}$OT can be arbitrarily made and later reversed.

The price of inflexibility in the OT case is a need for additional prior transfers. Given that the arrival of a bit is an *irrevocable* event, it is remarkable that such flexibility can be accommodated at all. We show that these additional transfers are, in fact, necessary.

2 Preliminaries and Definitions

We use "$c \leftarrow \$$" to denote assigning c the value 0 or 1 with equal probability. Local variables are sometimes referred to in a (C++)-like notation to distinguish them from similarly-named local variables held by another agent: for example, $P : x$ denotes P's local variable x. The notation "$P \rightarrow Q : x$" denotes the event that P sends $P{:}x$ to Q. The notation "$Q \leftarrow P : y$" denotes the event that Q receives a message from P and stores it as $Q{:}y$. (We sometimes omit explicit mention of message reception.)

For clarity, we ignore blatantly detectable misbehavior (*e.g.* sending syntactically incorrect messages) by assuming that an honest party rewards it by refusing further interaction with the offender. Including such non-instructive cases makes for unreadable code.

The reader well-versed in OT may skip to §2.3 on first reading.

2.1 The Variants of Oblivious Transfer

We describe oblivious transfer formally in terms of a three-party protocol involving Alice, Bob, and an absolutely trusted third party, communicating over synchronized and absolutely secure channels. Essentially, the third party represents the desired primitive, and the protocol describes how the primitive is used. Let $\hat{A}$ be Alice and $\hat{B}$ be Bob; then our specification of Rabin's original concept would be the following protocol for for $\hat{A}$, $\hat{B}$, and third party OT:

Oblivious Transfer		
$\hat{A}$:	input b	
$\hat{A} \rightarrow$ OT:	b	
OT:	$?b \leftarrow \$$	
OT $\rightarrow \hat{B}$:	$(?b, ?b \cdot b)$	//either $(0,0)$ or $(1,b)$

The value of $?b$ records whether Bob receives a valid bit or not. The other standard variations on OT involve the special third parties $\frac{1}{2}$OT and $\binom{2}{1}$OT:

One-Out-Of-Two Oblivious Transfer	
$\hat{A}$:	input b_0, b_1
$\hat{A} \rightarrow \frac{1}{2}$OT:	(b_0, b_1)
$\frac{1}{2}$OT:	$c \leftarrow \$$
$\frac{1}{2}$OT $\rightarrow \hat{B}$:	b_c

Chosen One-Out-Of-Two Oblivious Transfer	
$\hat{A}$:	input b_0, b_1
$\hat{B}$:	input c
$\hat{A} \rightarrow \binom{2}{1}$OT:	(b_0, b_1)
$\hat{B} \rightarrow \binom{2}{1}$OT:	c
$\binom{2}{1}$OT $\rightarrow \hat{B}$:	(c, b_c)

2.2 Secure Implementations

A full formal framework with details of the tedious proofs included in our submitted version is omitted in this presentation – undoubtedly to the relief of many. Details will appear again in the final version. We sketch some of the main aspects, but the reader is invited to skip to §2.3.

In particular, we would like to map an *implementation*, namely a three-party protocol, $\langle A(x_a, k), B(x_b, k), P(k)\rangle$, to a *specification*, namely another three-party protocol, $\langle \hat{A}(x_a, k), \hat{B}(x_b, k), \hat{P}(k)\rangle$, and show that the implementation achieves results asymptotically indistinguishable from the specification (as $k \rightarrow \infty$). Here, x_a and x_b are the inputs of Alice and Bob, and P suggests a *primitive*, namely a trusted third party such as OT, $\frac{1}{2}$OT, or $\binom{2}{1}$OT.

The implementation should model the results of the specification even when attacked by an adversary, **adv**. In particular, attacks on the implementation should be mapped to "equivalent" attacks on the specification. We provide such a mapping by way of an *interface* that envelops **adv**, converting **adv**'s attacks to equivalent actions in the specification protocol, and converting the information gained in the specification protocol to appear as though it came from the implementation, for **adv**'s sake.

Let k be a security parameter. When **adv** attacks implementation $\langle A(x_a, k), B(x_b, k), P(k)\rangle$, let $[\mathbf{adv}, A, B, P]$ denote the distribution on the inputs/outputs $((y_{\mathbf{adv}}, k), (x_A, y_A, k), (x_B, y_B, k))$ of **adv**, A and B, respectively.

Definition 1. Protocol $\langle A, B, P\rangle$ is a $(1 - O(2^{-k}))$-**secure implementation** of protocol $\langle \hat{A}, \hat{B}, \hat{P}\rangle$ if there exists an interface $\mathcal{I}$ such that for all adversaries **adv**, for all $x_{\mathbf{adv}}$, x_a and x_b,

$$[\mathbf{adv}, A, B, P] \approx^{O(2^{-k})} [\mathcal{I}(\mathbf{adv}), \hat{A}, \hat{B}, \hat{P}].$$

We consider *adaptive adversaries*, namely adversaries who can corrupt any number of players (except trusted primitives) at any point during the protocol.

The details of the interfaces will not appear here (the various sequences of events it must handle are numerous, particularly when handling adaptive adversaries), but we remark on two important aspects.

First, when the specification can be shown information-theoretically private (perfectly or statistically), the job of the interface reduces to supplying "faked" samples to **adv**, using a distribution that is independent of the inputs to which the interface has no access (namely, those held by uncorrupted players). For the protocols we have designed, this simulation is easy to do in polynomial time, typically consisting of mere coin flips or exclusive-or's with earlier messages.

Second, for the protocols presented here, the interface must convert messages from **adv** (who plays the part of a corrupt A or B) into "equivalent" messages sent by a corrupt $\hat{A}$ or $\hat{B}$ to the trusted third party, $\hat{P}$. The interface does so without looking "inside" the adversary, but rather by performing simple computations on the messages **adv** sends. (Subtly but importantly, this demonstrates that **adv** "knows" what effective messages it sends [Bea92].) We thus avoid any need to derive an "intent" or a "strategy" of the adversary in order to achieve the same effects when attacking the secure specification protocol.

2.3 Precomputed Reductions

Let $(\hat{A}, \hat{B}, \hat{P})$ be a specification in which $\hat{A}$ and $\hat{B}$ invoke a primitive, $\hat{P}$, precisely once, and then output their results. The three OT protocols listed in §2.1 are such protocols.

A **precomputed reduction** from one primitive, $\hat{P}$, to another primitive, P, is a two-stage protocol (A, B, P) that implements $(\hat{A}, \hat{B}, \hat{P})$, $O(2^{-k})$-securely. In the first, or *precomputation* stage, A and B can invoke P freely. At the start of the second, or *online* stage, A and B receive their respective inputs x_a and x_b, but no further communication with P is possible. If there is a precomputed reduction from $\hat{P}$ to P, we write $\hat{P} \overset{\text{pre}}{\leq} P$.

An **online** reduction has no precomputation stage. If there is an online reduction from $\hat{P}$ to P, we write $\hat{P} \overset{\text{onl}}{\leq} P$.

The **complexity** of an implementation is the worst-case number of times the protocol invokes P.

3 Precomputed $\frac{1}{2}$OT or $\binom{2}{1}$OT

We first describe implementations of all three variants using a single precomputed $\frac{1}{2}$OT (or a single precomputed $\binom{2}{1}$OT).

3.1 Precomputed Reduction: $\frac{1}{2}$OT to $\frac{1}{2}$OT

Figure 2 presents the simplest and cheapest precomputed reduction, $\frac{1}{2}\text{OT} \overset{\text{pre}}{\leq} \frac{1}{2}\text{OT}$, which raises some subtle issues despite its simplicity. Roughly speaking, Alice initially transfers two random bits, r_0 and r_1. Bob receives one of them,

$\frac{1}{2}\mathrm{OT} \overset{\text{pre}}{\leq} \frac{1}{2}\mathrm{OT}$

Precomputation:		*Online Stage:*	
A:	$r_0 \leftarrow \$,\ r_1 \leftarrow \$$	A:	input b_0, b_1
$A \rightarrow \frac{1}{2}\mathrm{OT}$:	(r_0, r_1)		$c \leftarrow \$$
$\frac{1}{2}\mathrm{OT}$:	$d \leftarrow \$$	$A \rightarrow B$:	$(c,\ b_0 \oplus r_c,\ b_1 \oplus r_{\overline{c}})$
$\frac{1}{2}\mathrm{OT} \rightarrow B$:	(d, r_d)	B:	receive (c, x_0, x_1)
			output $(d \oplus c,\ r_d \oplus x_{d \oplus c})$

Fig. 2. Precomputed self-reduction for $\frac{1}{2}\mathrm{OT}$.

r_d, and knows which one it is. Later, when Alice has decided on b_0 and b_1, she sends the corrections $(b_0 \oplus r_0, b_1 \oplus r_1)$ to Bob. More accurately, Alice flips a coin, c, and may instead send the corrections $(b_0 \oplus r_1, b_1 \oplus r_0)$ to Bob. Bob thus gets $(b_0 \oplus r_c, b_1 \oplus r_{\overline{c}})$, and, knowing r_d, he can calculate $b_{d \oplus c}$. The other bit, $b_{\overline{d \oplus c}}$, remains masked by the unknown value $r_{\overline{d}}$.

The inclusion of c in this reduction is a very subtle point. If omitted, which one may be apt to do without careful consideration, then Bob learns in advance which bit he will receive – b_0 or b_1 – even though he does not learn the bit's value until the online stage.

This leak seems innocuous at first. But say that Bob the gambler is awaiting the answer b_0 to the question, "Will the roulette wheel in Las Vegas give black or red at 12:00 noon?" or the answer b_1 to the question, "Will the roulette wheel in Monte Carlo give black or red at 12:00 noon?" He has an inside contact in each casino who will let him know the local answer at 11:55, but one of those insiders will surely end up nabbed by the police (or other unsundry enforcers). At best, Bob can wait either in Las Vegas or in Monte Carlo and hope that the insider at his chosen location can give him the answer. If Bob doesn't know which insider will survive, he has at best a 50-50 chance of getting the prediction. But if Bob does find out which insider will survive, he simply waits in the right casino and is guaranteed an easy profit.

Indeed, without c, this reduction bears some similarity to the "non-interactive" OT implemented in [BM89, HL90]. (Such methods employ complexity-theoretic assumptions to transfer sequences of bits, and seek weaker[2] goals for the purpose of supporting particular tasks such as non-interactive proof systems.) If such a coin flip had been included, the non-interactive zero-knowledge proof system of [BM89] would have been secure against the very subtle attack proposed in [CHL94]. In contrast, the protocol described here is provably secure:

Theorem 2. *$\frac{1}{2}OT \overset{\text{pre}}{\leq} \frac{1}{2}OT$ with complexity 1. That is, $\frac{1}{2}OT$ can be implemented using 1 prior invocation of $\frac{1}{2}OT$.*

[2] Bob's reception of each successive bit is not independent of his reception of earlier bits. Alice may derive correlations in reception patterns, although she does not learn the precise patterns themselves.

Proof Sketch. Our goal is to convert attacks on the implementation (fig. 2) into equally-powered attacks on the specification (§2.1).

In the precomputation stage, Alice receives nothing. If **adv** corrupts her, we (the interface/simulator $\mathcal{I}$) simply keep track of what **adv** generates on her behalf. Bob receives the pair (d, r_d), which we need to generate for him only if **adv** has corrupted him. If so, we generate a uniformly random (d, r_d).

In the online stage, if **adv** chooses to corrupt Alice, we choose to corrupt $\hat{A}$ (the Alice in the specification protocol), thereby obtaining inputs b_0 and b_1. We generate a random r_0 and r_1, then hand r_0, r_1, b_0, b_1 to **adv** as the "faked" view of Alice. (If Alice was already corrupted in the precomputation stage, then we just hold onto the (r_0, r_1) she gave us earlier, hand her b_0, b_1, and continue as described below.)

Then, **adv** generates (c, x_0, x_1) on faulty Alice's behalf. We calculate Alice's *effective* bits as $\beta_0 = x_0 \oplus r_c$, $\beta_1 = x_1 \oplus r_{\bar{c}}$. (These bits may or may not be b_0, b_1. If **adv** is merely curious but honest, they indeed will be b_0, b_1. But whatever clever or mysterious methods **adv** uses to come up with (c, x_0, x_1), we can easily determine what the message really "means," *in terms of messages carrying equivalent information and influence in the specification.*) Having corrupted $\hat{A}$ in the specification protocol (see §2.1), we send (β_0, β_1) to $\frac{1}{2}$OT on behalf of $\hat{A}$. As a result, $\hat{B}$ will output either β_0 or β_1, which is exactly what Bob will do in the implementation. (Technically, let $\hat{\gamma}$ be $\frac{1}{2}$OT:c in the specification and let γ be **adv**:$c \oplus \frac{1}{2}$OT:d in the implementation. Then strictly speaking, $\hat{B}$ outputs $(\hat{\gamma}, \beta_{\hat{\gamma}})$ while Bob outputs (γ, β_γ), but in either scenario, $\hat{\gamma}$ or γ is uniformly random and fully independent of $\hat{A}$/Alice's view.)

The case in which **adv** corrupts Bob is simple, because Bob's role is essentially passive. All we need do is to corrupt $\hat{B}$ and discover the value (D, b_D) that the online $\frac{1}{2}$OT sends on to $\hat{B}$ in the specification. We don't need to discover the value $b_{\overline{D}}$, a fact which incidentally demonstrates that the implementation does not leak the value of the unreceived bit. Using the faked, uniformly random values (d, r_d), we set $c = d \oplus D$ (to make $D = d \oplus c$ be the index of the received bit), set $x_D = b_D$, and choose $x_{\bar{D}}$ at random. Finally, we hand Bob the triple (c, x_0, x_1). A curious but law-abiding adversary might then calculate Bob's output as $b_D = x_D$. In any case, a corrupt Bob receives precisely the same information in the implementation as in the specification.

This outline of an interface-based mapping can be filled in to show formally that any adversarial attack on the implementation can be equated with an attack on the specification, with perfectly identical distributions (on inputs/outputs of honest players and views of adversaries). □

3.2 Precomputed Reduction: $\binom{2}{1}$OT to $\frac{1}{2}$OT

Reducing $\binom{2}{1}$OT to precomputed $\frac{1}{2}$OT is slightly more complicated, particularly since is it somewhat surprising that a chosen reception can be obtained using a random reception.

$\binom{2}{1}\text{OT} \overset{\text{pre}}{\leq} \frac{1}{2}\text{OT}$

Precomputation:		*Online Stage:*	
A:	$r_0 \leftarrow \$,\ r_1 \leftarrow \$$	A:	input b_0, b_1
$A \rightarrow \frac{1}{2}\text{OT}$:	(r_0, r_1)	B:	input c
$\frac{1}{2}\text{OT}$:	$d \leftarrow \$$	$B \rightarrow A$:	$e = c \oplus d$
$\frac{1}{2}\text{OT} \rightarrow B$:	(d, r_d)	$A \rightarrow B$:	$(b_0 \oplus r_e, b_1 \oplus r_{\overline{e}})$
		$B \leftarrow A$:	(x_0, x_1)
		B:	output $x_c \oplus r_d$

Fig. 3. Precomputed reduction from $\binom{2}{1}\text{OT}$ to $\frac{1}{2}\text{OT}$.

Fig. 3 describes the implementation. Again, Alice initially transfers two random bits r_0 and r_1 via the initial $\frac{1}{2}$OT. Bob receives one of them, namely r_d, and he knows which one he received.

Later, after Alice has chosen her desired bits b_0 and b_1, she sends corrections like those in the $\frac{1}{2}\text{OT} \overset{\text{pre}}{\leq} \frac{1}{2}\text{OT}$ protocol. This time, Bob selects which sort of correction she should send, for otherwise he would have no control over which b_i he received. In particular, if Bob wants to discover b_c, he sends Alice $e = c \oplus d$. She provides him with $(b_0 \oplus r_e, b_1 \oplus r_{\overline{e}})$, and using r_d, Bob can decode $b_{e \oplus d}$, namely b_c. The other bit, $b_{\overline{c}}$, remains masked by the unknown bit $r_{\overline{d}}$.

The question remains whether Alice learns anything from the message e sent by Bob. From Alice's point of view, if Bob requests $e = 0$, there is an equal chance that $d = 0$ and Bob wants b_0 or that $d = 1$ and Bob wants b_1. Likewise, if Bob requests $e = 1$, there is an equal chance that $d = 0$ and Bob wants b_1 or that $d = 1$ and Bob wants b_0. The secret random bit d provided by the $\frac{1}{2}$OT primitive is enough to protect Bob's privacy.

Theorem 3. $\binom{2}{1}OT \overset{\text{pre}}{\leq} \frac{1}{2}OT$ *with complexity* 1. *That is,* $\binom{2}{1}OT$ *can be implemented using* 1 *prior invocation of* $\frac{1}{2}OT$.

Proof. Similar to the proof of Theorem 2. As before, a key ingredient is the extraction of Alice's *effective* bits by computing $(x_0 \oplus r_e, x_1 \oplus r_{\overline{e}})$ (which are certainly (b_0, b_1) if Alice is honest) and handing these to $\binom{2}{1}$OT (the primitive available in the specification protocol), so that $\hat{B}$ sees the same distribution on chosen bits as adv induces Bob to calculate in the implementation. Another key ingredient consists of randomly faking (r_0, r_1) and e for B, and later compensating for this by faking the later message (x_0, x_1) with an appropriate distribution, even though we have access to only *one* of Alice's bits (by way of $\binom{2}{1}$OT). □

3.3 Other Precomputed Reductions

To facilitate the discussion of other precomputations, we first consider a few online reductions. While online reductions from OT to $\frac{1}{2}$OT or to $\binom{2}{1}$OT are trivial, an online reduction from $\frac{1}{2}$OT to $\binom{2}{1}$OT requires care, despite its simplicity.

Lemma 4. *$\frac{1}{2}OT \overset{\text{onl}}{\leq} \binom{2}{1}OT$, with complexity 1. That is, there is an online reduction from $\frac{1}{2}OT$ to $\binom{2}{1}OT$ that invokes $\binom{2}{1}OT$ precisely once.*

Proof Sketch. First, Alice simply switches b_0 and b_1 randomly (by transferring $(b_d, b_{\overline{d}})$ for $d \leftarrow \$$), before allowing Bob to choose. Second, Bob makes a *random* choice $c \leftarrow \$$, thereby obtaining $b_{d \oplus c}$.

The subtle aspect is in requiring Bob to make a random choice. Clearly, there is an unsubtle privacy issue: Alice must be prevented from learning which bit Bob gets. More subtly, however, there is a correctness issue: Alice must not be able to cause Bob to receive a deterministic selection of bits. The random coin toss of the $\frac{1}{2}$OT primitive is effectively the exclusive-or of Alice's reversal choice with Bob's choice, and if Bob's choice is not random, the implementation will not correspond to the specification. In order for honest Bob to get a random selection – which could conceivably be crucial to some higher-level protocol[3] – he must randomize the choice himself, to protect against Alice's failing to do so.

(As a side remark, we note that this aspect becomes obvious when one tries to find an interface-based comparison between implementation and specification. In particular, the distribution obtained by honest $\hat{B}$ in the specification always contains the uniformly random c chosen by $\binom{2}{1}$OT. If Bob's random choice in the implementation is overlooked, then c is whatever Alice sent him, and the implementation's results are easily distinguished from the specification's results.) □

Online reductions facilitate the development of precomputed reductions:

Lemma 5. *Let P, Q and R be (not necessarily distinct) primitives from $\{\text{OT}, \frac{1}{2}OT, \binom{2}{1}OT\}$. Then:*

(A) If $P \overset{\text{onl}}{\leq} Q$ and $Q \overset{\text{pre}}{\leq} R$ then $P \overset{\text{pre}}{\leq} R$.
(B) If $P \overset{\text{pre}}{\leq} Q$ and $Q \overset{\text{onl}}{\leq} R$ then $P \overset{\text{pre}}{\leq} R$.

Proof. (A) follows from incorporating the reduction $P \overset{\text{onl}}{\leq} Q$ into the online stage of the precomputed reduction demonstrating $Q \overset{\text{pre}}{\leq} R$. (B) follows from incorporating the reduction $Q \overset{\text{onl}}{\leq} R$ into the precomputation stage of the precomputed reduction demonstrating $P \overset{\text{pre}}{\leq} Q$. □

We can now start to fill in the columns of Fig. 1.

Lemma 6. *$\text{OT} \overset{\text{pre}}{\leq} \frac{1}{2}OT$, $\frac{1}{2}OT \overset{\text{pre}}{\leq} \frac{1}{2}OT$, and $\binom{2}{1}OT \overset{\text{pre}}{\leq} \frac{1}{2}OT$, each with complexity 1.*

[3] Imagine a protocol in which Bob makes use of the c's he obtains from $\frac{1}{2}$OT to run a primality test, for example. While this may be odd, stupid, and contrived, it is a mathematically acceptable use of the results of the $\frac{1}{2}$OT primitive.

Proof. The first two cases follow from Theorems 2 and 3. Applying lemma 5 to Theorem 2 and the trivial online reduction $\mathrm{OT} \overset{\text{onl}}{\leq} \frac{1}{2}\mathrm{OT}$ gives the third. Each reduction (whether online or precomputed) requires only one invocation of its respective primitive. □

Lemma 7. $\mathrm{OT} \overset{\text{pre}}{\leq} \binom{2}{1}OT$, $\binom{2}{1}OT \overset{\text{pre}}{\leq} \binom{2}{1}OT$, *and* $\frac{1}{2}OT \overset{\text{pre}}{\leq} \binom{2}{1}OT$, *each with complexity* 1.

Proof. Using lemmas 6, 4, and 5,

$$\begin{aligned}
\mathrm{OT} \overset{\text{pre}}{\leq} \tfrac{1}{2}\mathrm{OT} \overset{\text{onl}}{\leq} \tbinom{2}{1}\mathrm{OT} \Rightarrow \quad & \mathrm{OT} \overset{\text{pre}}{\leq} \tbinom{2}{1}\mathrm{OT} \\
\tfrac{1}{2}\mathrm{OT} \overset{\text{pre}}{\leq} \tfrac{1}{2}\mathrm{OT} \overset{\text{onl}}{\leq} \tbinom{2}{1}\mathrm{OT} \Rightarrow \quad & \tfrac{1}{2}\mathrm{OT} \overset{\text{pre}}{\leq} \tbinom{2}{1}\mathrm{OT} \\
\tbinom{2}{1}\mathrm{OT} \overset{\text{pre}}{\leq} \tfrac{1}{2}\mathrm{OT} \overset{\text{onl}}{\leq} \tbinom{2}{1}\mathrm{OT} \Rightarrow \quad & \tbinom{2}{1}\mathrm{OT} \overset{\text{pre}}{\leq} \tbinom{2}{1}\mathrm{OT}
\end{aligned}$$

□

4 Precomputed OT

We now turn to the precomputed reduction from OT to itself. The precomputed reductions from $\frac{1}{2}$OT and $\binom{2}{1}$OT to OT are similar in content and cost to this preprocessed self-reduction.

Figure 4 shows a preprocessed reduction from the standard OT to itself, using $15k = \Theta(k)$ invocations of OT in the preprocessing stage. The protocol is adapted from Crépeau's online reduction [Cré87] from $\frac{1}{2}$OT to OT. (The necessity of using $\Omega(k)$ OT's is argued below; see Theorem 9.) In these $15k$ transfers, Bob will receive between $5k+1$ and $10k-1$ bits, with probability exceeding $1-O(2^{-k})$. He can thus find two disjoint $5k$-sets U_0 and U_1 containing indices of received and unreceived bits, respectively. He can't find two *disjoint* sets containing only *received* bits, however. Alice uses these sets of indices to construct two masks, v_0 and v_1, by taking the exclusive-or over all r_i for a given set. Having received everything indexed by U_0, Bob can calculate either v_0 or v_1, but not both. In the online stage, Alice sends b, masked with either v_0 or v_1, randomly, giving Bob a 50-50 chance to calculate b. Since Alice does not know which of the two sets U_0 and U_1 is the set that Bob received, she does not know whether Bob receives b.

Theorem 8. $\mathrm{OT} \overset{\text{pre}}{\leq} \mathrm{OT}$ *with complexity* $\Theta(k)$. *That is,* OT *can be implemented using* $\Theta(k)$ *prior invocations of* OT.

Proof Sketch. The proof is similar to that of Theorem 2, but complicated by the (negligible) possibility of error. Conditioned on the event that the trusted OT party transmits between $5k+1$ and $10k-1$ bits to Bob, the interface-mapped attacks on the specification (§2.1) achieve results identical to those obtained by attacks on the implementation (Fig. 4). The chance that this event

$\mathrm{OT} \overset{\text{pre}}{\leq} \mathrm{OT}$

Precomputation:			$A \leftarrow B$:	(V_0, V_1)	
A:	$r_i \leftarrow \$$	$(i = 1..15k)$	A:	reject if $V_0 \cap V_1 \neq \emptyset$	
$A \rightarrow \mathrm{OT}_i$:	r_i	$(i = 1..15k)$		$v_0 = \oplus_{i \in V_0} r_i$	
$\mathrm{OT}_i \rightarrow B$:	$(?r_i, r_i)$	$(i = 1..15k)$		$v_1 = \oplus_{i \in V_1} r_i$	
B:	$R = \{i \mid ?r_i = 1\}$		*Online Stage:*		
	Choose random $U_0 \subseteq R$, $\lvert U_0 \rvert = 5k$		A:	input b	
	Choose random $U_1 \subseteq \overline{R}$, $\lvert U_1 \rvert = 5k$			$d \leftarrow \$$	
	$u = \oplus_{i \in U_0} r_i$		$A \rightarrow B$:	$(d, b \oplus v_d)$	
	$f \leftarrow \$$		B:	receive (d, x)	
$B \rightarrow A$:	$(U_f, U_{\overline{f}})$			if $d = f$ then	
				output $(1, u \oplus x)$	
				else output $(0, 0)$	

Fig. 4. Precomputed self-reduction for OT.

fails to occur is at most $10k[15k!/(10k!)(5k!)]2^{-15k}$. Using Stirling's formula, $15k!/(10k!)(5k!) \leq e^{9.55k} \leq 2^{13.8k}$. Thus the chance that the event fails is bounded by $10k \cdot 2^{13.8k} 2^{-15k} = O(2^{-k})$. Removing the condition that this event occurs, we see that the results are $O(2^{-k})$-indistinguishable. □

Theorem 9. *Any precomputed reduction from* OT *to* OT *that is* $(1 - O(2^{-k}))$*-secure against static* 1*-adversaries (even honest-but-curious ones) must use* $\Omega(k)$ *prior transfers.*

Proof. Assume there exists a $(1 - 2^{-k})$-secure implementation that does not use $\Omega(k)$ transfers. Then for infinitely many k, at most $k/10$ transfers are used. We create a static 1-adversary **adv** that corrupts Bob at the outset, but **adv** directs Bob to follow his program precisely. With probability at least $2^{-k/10}$, Bob receives *all* bits in the precomputation stage, in which case **adv** predicts with virtually 100% accuracy that it will indeed discover the b sent later on. In the specification protocol for OT, **adv** has no such ability to predict whether Alice's actual bit will arrive, and the $2^{-k/10}$-correlation of **adv**'s prediction with the reception of Alice's bit is enough to show that the protocol is not 2^{-k}-secure.

In slightly more detail, consider random variables describing Alice's internal random choices (α), Bob's (β), the OT reception pattern ($R \in \{0,1\}^{k/10}$, with a "1" indicating reception), and the conversation (C) in the precomputation stage. Call C *fair* if $\Pr[\text{Bob outputs } (1,b) \text{ later} | C] > .49$ and $\Pr[\text{Bob outputs } (0,0) \text{ later} | C] > .49$. Then $\Pr[C \text{ unfair}] \leq 2^{-k}$, else the protocol is not $(1 - 2^{-k})$-secure (since even an honest Alice would learn the fate of her bit). Furthermore, $\Pr[C \text{ unfair} | R = 111\cdots1] \leq 2^{-k/2}$, else $\Pr[C \text{ unfair}] > 2^{-k/10} 2^{-k/2} > 2^{-k}$.

Now, **adv**'s only unusual behavior is to make a prediction when it sees $R = 111\cdots1$ (*ie.* Bob received all OT's). Even if Bob outputs $(0,0)$ (Bob is law-abiding, after all), with high probability C is fair and **adv** can inspect Bob's

records to determine a different β, consistent with C, that would have caused Bob to output $(1, b)$. (Naturally, the tiny error rate does admit the possibility of β's causing Bob to output $(1, \bar{b})$. If C is fair, the probability of this is at most .02.)

Thus, not only does **adv** obtain Alice's bit, but it is able to predict in advance that it will obtain the bit. In particular, in the assumed implementation, **adv**'s advantage is at least the following (fair and correct, minus fair and incorrect, minus unfair):

$$2^{-k/10}(1-2^{-k/2})(.98) - 2^{-k/10}(1-2^{-k/2})(.02) - 2^{-k/10}(2^{-k/2}).$$

This exceeds the best-possible 0 advantage in the specification protocol by far more than 2^{-k}, thus the implementation is not $(1-2^{-k})$-secure. It is a simple matter to extend $1-2^{-k}$ to $1-O(2^{-k})$. □

The proof generalizes to show that $\omega(\log k)$ transfers are necessary to achieve $(1-k^{-\omega(1)})$ security (*i.e.* "statistical" security), and in particular, that using $O(1)$ transfers admits a constant probability of error. It also implies lower bounds for *online* reductions from $\frac{1}{2}$OT or $\binom{2}{1}$OT to OT, showing that Crépeau's methods are asymptotically optimal [Cré87].

5 Conclusions

Putting together Lemmas 6 and 7 and Theorems 8 and 9, we conclude:

Theorem 10. *All variants of oblivious transfer can be reduced to precomputed invocation(s) of any variant, with tight complexity bounds as given in the table in Figure 1.*

In particular, one-out-of-two oblivious transfer can be performed in advance and used to implement any variant of OT later on, at a cost of one prior transfer per future transfer. Chosen one-out-of-two OT also suffices. Unfortunately, Rabin's original OT cannot be used in this manner, as it requires $\Omega(k)$ prior tranfers per future transfer.

These reductions are information-theoretically secure and permit prior execution of oblivious transfers, whether the execution involves quantum channels, Blum integers, or any other construct.

It is not hard to see that the reductions described in §3 and §4 apply to string transfers, letting variables b, b_0 and b_1 represent strings and interpreting the $\oplus$ symbol as bitwise exclusive-or, as appropriate.

Moreover, the discrete-log based "non-interactive" oblivious transfer channels of [BM89, HL90, Har91] can now be used correctly to implement *any* variant of OT, at a cost of 1 bit per transfer. The proof of this claim requires significantly greater formalism to address computational security and is beyond the scope of this work.

References

[Bea92] D. Beaver. "How to Break a 'Secure' Oblivious Transfer Protocol." *Advances in Cryptology – Eurocrypt '92 Proceedings*, Springer–Verlag LNCS 658, 1993, 285–296.

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

[BBCS91] C. Bennett, G. Brassard, C. Crépeau, M. Skubiszewska. "Practical Quantum Oblivious Transfer." *Advances in Cryptology – Crypto '91 Proceedings*, Springer–Verlag LNCS 576, 1992, 351–366.

[Boe91] B. den Boer. "Oblivious Transfer Protecting Secrecy." *Advances in Cryptology – Eurocrypt '91 Proceedings*, Springer–Verlag LNCS 547, 1991, 31–45.

[BC90] G. Brassard, C. Crépeau. "Quantum Bit Commitment and Coin Tossing Protocols." *Advances in Cryptology – Crypto '90 Proceedings*, Springer–Verlag LNCS 537, 1991, 49–61.

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

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

[CHL94] Y-H. Chen, T. Hwang, C-M. Li. "On the Zero Knowledge Proof Systems Based on One-out-of-two Non-Interactive Oblivious Transfers." Manuscript, 1994.

[Cré87] C. Crépeau. "Equivalence Between Two Flavours of Oblivious Transfers." *Advances in Cryptology – Crypto '87 Proceedings*, Springer–Verlag LNCS 293, 1988, 350–354.

[CK88] C. Crépeau, J. Kilian. "Achieving Oblivious Transfer Using Weakened Security Assumptions." *Proceedings of the* 29^{th} *FOCS*, IEEE, 1988, 42–52.

[EGL82] S. Even, O. Goldreich, A. Lempel. "A Randomized Protocol for Signing Contracts." *Proceedings of Crypto 1982*, Springer–Verlag, 1983, 205–210.

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

[Har91] L. Harn. "An Oblivious Transfer Protocol and Its Application for the Exchange of Secrets." *Advances in Cryptology – Asiacrypt '91 Proceedings*, Springer-Verlag LNCS 739, 1993, 312–320.

[Kil88] J. Kilian. "Founding Cryptography on Oblivious Transfer." *Proceedings of the* 20^{th} *STOC*, ACM, 1988, 20–29.

[Rab81] M. Rabin. "How to Exchange Secrets by Oblivious Transfer." TR-81, Harvard, 1981.