Concurrent Oblivious Transfer

JUAN A. GARAY PHILIP MACKENZIE
Bell Laboratories – Lucent Technologies
600 Mountain Ave.
Murray Hill, NJ 07974, USA
{garay,philmac}@research.bell-labs.com

Abstract

We consider the problem of designing an efficient oblivious transfer (OT) protocol that is provably secure in a concurrent setting, i.e., where many OT sessions may be running concurrently with their messages interleaved arbitrarily. Known OT protocols use zero-knowledge proofs, and no concurrent zero-knowledge proofs are known that use less than a poly-logarithmic number of rounds (at least without requiring a pre-processing phase, a public random string, an auxiliary string, timing constraints, or pre-distributed public keys). We introduce a model for proving security of concurrent OT protocols, and present a protocol that is proven secure in this model based on the Decisional Diffie-Hellman problem, assuming inputs for the respective parties and for different protocol instances are chosen independently. The protocol is efficient, requiring only a slightly non-constant number of rounds.

1. Introduction

Oblivious Transfer (OT) is a fundamental cryptographic primitive introduced by Rabin [45]. Informally, an OT protocol is a two-party protocol in which Alice transmits a bit b to Bob, Bob receives the bit with probability $\frac{1}{2}$, and Alice does not learn whether Bob receives the bit. A variant of the problem, called $\binom{2}{1}$-OT, has Alice transmitting two bits (b_0, b_1) to Bob, Bob choosing which bit b_e to receive, Alice not learning e, and Bob not learning b_{1-e}. OT, $\binom{2}{1}$-OT, and other variants have been studied extensively (e.g., [1, 2, 3, 5, 10, 9, 11, 22, 48, 19, 25, 30, 39, 37]) and have been used in many cryptographic protocols (e.g., [38, 43, 8]).

Beaver [2] was the first to formally define security for OT in a *stand-alone* setting (hereafter referred to as *stand-alone OT*). He defined it in relation to an ideal system for OT in which Alice and Bob communicate through a trusted host that performs the OT operation, and in which all communication to and from the trusted host is absolutely private. To prove a real OT protocol is secure, one must be able to simulate the real protocol using an ideal OT protocol. The idea is that an adversary that breaks the real protocol can be used to create an adversary that breaks the ideal protocol, which by definition is impossible.

Concurrent composition of protocols. When constructing cryptographic protocols for the real world, and in particular, when they are meant to be run over the Internet, they need to be secure even when instances of the protocols are run concurrently. There has been a great deal of work on securing authentication and key exchange protocols in a concurrent setting (e.g., [4, 6, 7, 24, 35, 36, 42, 44, 49]).[1]

There has also been recent work on zero-knowledge protocols in a concurrent setting. Dwork, Naor and Sahai [27] considered the general case of concurrent zero knowledge in an asynchronous setting, and showed that assuming bounds on the relative speed of the processors, one can construct four-round zero-knowledge arguments for any language in NP. This result was improved by Dwork and Sahai [28], who showed that the timing assumption can be reduced to a pre-processing phase. The timing assumptions were later eliminated by Di Crescenzo and Ostrovsky [18], who presented a constant-round concurrent zero-knowledge protocol (proofs and arguments) with pre-processing for all languages in NP. Recently, Damgård [21] presented constant-round concurrent zero-knowledge protocols assuming an auxiliary string is available to all players (see [21]), and Canetti *et al.* [12] presented constant-round concurrent zero-knowledge protocols in the public-key model.

With no pre-processing or timing assumptions—i.e., the "vanilla" model, Kilian, Petrank and Rackoff [41] showed that if a language L has a four-round proof-of-membership, and one can black-box simulate polynomially many asynchronous concurrent proofs, then $L \in$ BPP. The number of

[1] This work was preceded by many attacks on previous protocols that were not designed with careful consideration of the concurrent setting.

rounds in this lower bound has recently been increased to seven by Rosen [47]. Regarding upper bounds, Richardson and Kilian [46] presented a concurrent zero-knowledge proof for this setting but they required a polynomial number of rounds (polynomial in the concurrency). More recently, Kilian and Petrank [40] have reduced the number of rounds to poly-logarithmic.

To our knowledge, there has been no previous work on OT in a concurrent setting (hereafter referred to as *concurrent OT*). On the other hand, there have been protocols which are meant to be run in a concurrent setting and which are based on OT [43, 8]. In these papers, OT is essentially assumed to be a black-box in which concurrently-running protocols have no effect on each other. However, all OT protocols in the literature have used zero-knowledge proofs, which, as discussed above, may not necessarily be secure when run concurrently! In fact, the only ones shown to be secure require at least a poly-logarithmic number of rounds (in the concurrency), and thus would be very costly in practice.

Our results. In this paper we focus on $\binom{2}{1}$-OT.[2] We extend Beaver's stand-alone security model for OT [2] to $\binom{2}{1}$-OT in a concurrent setting. We then present a protocol for concurrent $\binom{2}{1}$-OT in the standard model (i.e., no timing constraints, pre-processing, auxiliary strings, etc.) and prove its security. Our protocol is based on a protocol due to Bellare and Micali [5], and is the first protocol to be proven secure without using zero-knowledge protocols (i.e., using only witness-hiding protocols). The security of our protocol is based on the Decisional Diffie-Hellman problem[3] and has a number of rounds (and number of exponentiations) which is only slightly non-constant. (Contrast this to the most efficient concurrent zero-knowledge protocol known which requires a poly-logarithmic number of rounds.) Our proof requires that the input generation satisfies a certain *conditional simulatability* property (described in the next section), which is satisfied, for instance, if the inputs for the respective parties and for different protocol instances are chosen independently.

The technical contributions that underlie our results are as follows. A first contribution is to show an OT protocol that does not need to use a zero-knowledge protocol in order to be provably secure. This is not trivial, and in fact, the Bellare-Micali protocol that we extend is the only one that we have found that allows this. A major difficulty (especially in the concurrent setting), is to allow simulations without the zero-knowledge property of the ZK-protocols. That is, if a party must commit to certain values (or to the knowledge of certain values using a proof of knowledge), then our OT simulator must also, since the proof is not zero-knowledge.

A second contribution is the development of a new proof of representations, in which some part of the representation must be guaranteed to not be zero. This turns out to be very useful in our protocol, and may be useful in other contexts.

A third contribution is the development of proofs of knowledge that work in a concurrent setting, even if we cannot guarantee perfect indistinguishability between normal execution and execution after the extractor "rewinds."[4] We do this using a slightly non-constant number of standard three-round proofs of knowledge, and show that the extraction works with high probability for at least one of the standard proofs of knowledge.

2. Basic Cryptographic Concepts

We say that a function $f(n)$ is *negligible* if for any $poly(\cdot)$, there is an n' such that for $n > n'$, $f(n) < 1/poly(n)$.

Discrete logs and the DDH **problem.** For a prime q, we will work in a group G_q of order q with generator g. For a specific example, we could use a group formed by integer modular arithmetic as follows. Let $p = lq + 1$ be prime, and let Z_p^* be the multiplicative group modulo p, with g a generator of order q in Z_p^*. Then G_q would be the subgroup generated by g.

For $X = g^x$, we use $\mathrm{DL}(X, g) = x$ to denote the discrete log of X with respect to g. Additionally, for $Y = g^y$, let $\mathrm{DH}(X, Y) = g^{xy}$. We assume the hardness of the *Decision Diffie-Hellman problem* (DDH). In this problem, it is the goal of the polynomial-time adversary to distinguish the following two distributions with a non-negligible advantage over a random guess:

1. (X, Y, Z) with $X, Y, Z \in_R G_q$, and
2. $(X, Y, \mathrm{DH}(X, Y))$ with $X, Y \in_R G_q$.

Representations. We let $\mathrm{REP}(y, (g_1, g_2, \cdots, g_k))$ denote a representation of y over multiple bases/generators, i.e., $y = g_1^{u_1} g_2^{u_2} \cdots g_k^{u_k}$. In our protocols, we will need to prove knowledge of representations over multiple bases, but with the extra condition that the representation over one of them is non-zero. We use $\text{NZ-REP}_{g_i}(y, (g_1, g_2, \cdots, g_k))$ to denote a representation over the multiple bases, with the exponent on g_i being non-zero.

[2] Protocols for $\binom{2}{1}$-OT imply protocols for other variants of OT [15].

[3] Most known (provable) OT protocols are based on specific complexity assumptions; in [33] an OT protocol is given based on any trapdoor permutation.

[4] In particular, to avoid nested and interleaved rewinds in the concurrent setting, which could blow up the running time of the simulation, the extractor operates a rewind execution differently than the normal simulation operation.

3. Model and Definition of Secure OT

Our model used to prove security of $\binom{2}{1}$-OT in a concurrent setting is an extension of the model used for stand-alone OT by Beaver [2].

Security for $\binom{2}{1}$-OT is defined using an ideal system, which describes the service (of $\binom{2}{1}$-OT) that is to be provided, and a real system, which describes the world in which the protocol participants and adversaries work. The ideal system should be defined such that an "ideal world adversary" cannot (by definition) break the security. Then, intuitively, a proof of security would show that anything an adversary can do in the real system can also be done in the ideal system, and thus it would follow that the protocol is secure in the real system.

The ideal system. As described above, the ideal system described by Beaver for stand-alone OT includes two players, Alice and Bob, and a trusted host. Alice sends a bit b to the trusted host, who then sends $(1, b)$ or $(0, 0)$ to Bob with equal probability. Also, either player can send a `quit` message to the trusted host to abort the protocol. We assume all communication to or from the trusted host is absolutely private. Extending this to $\binom{2}{1}$-OT we would first have Bob send a bit e to the trusted host, then have Alice send two bits (b_0, b_1) to the trusted host, and finally have the trusted host send bit b_e to Bob. Security is considered with respect to a cheating Bob or a cheating Alice who tries to gain information that is not allowed.

We will assume the bits that Alice and Bob send to the trusted host are generated by a known probabilistic polynomial-time (PPT) Turing Machine $\mathcal{E}$, called the environment.[5]

The ideal system for concurrent $\binom{2}{1}$-OT has multiple Alice and Bob instances that send bits to, and receive bits from, the trusted host. Again the bits sent to the trusted host are assumed to be generated by an environment $\mathcal{E}$. Security is considered with respect to a set of cheating Bob instances or a set of cheating Alice instances.[6]

In both the stand-alone and the concurrent model of $\binom{2}{1}$-OT, once Bob receives a bit from the trusted host, both the index of the bit received and the bit itself is output. This models the adversary somehow obtaining these values, and is necessary for proving the security of these protocols in a "black-box" fashion so they can be used within larger protocols.

[5]Note that we specifically do not assume Alice or Bob has committed to any bits. This would be called "committed" or "verifiable" oblivious transfer [16, 17] and is beyond the scope of this paper.

[6]This is similar to the model for concurrent zero-knowledge, in which the adversary may corrupt either the provers (when considering soundness), or the verifiers (when considering zero-knowledge) but not some of each. See Section 6 for more discussion on this issue.

Because in the ideal system, (1) the sender in the ideal system cannot see the bit e sent by the receiver, and (2) the receiver cannot see the other bit b_{1-e} sent by the sender, it should be clear that the ideal $\binom{2}{1}$-OT is completely secure.

The real system. Now we (briefly) describe the real system in which we assume an OT protocol runs. As in the ideal system there will be multiple Alice and Bob instances. In the real system these are defined as state machines. Each instance starts in some initial state, and may transform its state based on its input bits (generated by the environment $\mathcal{E}$), and after that, only when it receives a message. At that point it updates its state and possibly sends a response message

As in the ideal system, security is considered with respect to a set of cheating Bob instances or a set of cheating Alice instances.

Conditional simulatability of $\mathcal{E}$ We say the environment $\mathcal{E}$ satisfies *conditional simulatability* for Alice (Bob) if for any set of inputs generated by $\mathcal{E}$ and their corresponding Alice (Bob) instances, conditioned on the values of those inputs, a PPT simulator may generate the inputs for all other Alice (Bob) instances such that they are perfectly indistinguishable from those produced by $\mathcal{E}$.[7]

Definition of security. Security for oblivious transfer is defined as follows:

1. *Completeness*: whenever any real world adversary faithfully delivers messages between an Alice and Bob instance the oblivious transfer completes successfully.
2. *Simulatability for Bob (Alice)*: for every PPT real world adversary $\mathcal{A}$ that corrupts a set of Alice (Bob) instances, there exists a PPT ideal world adversary $\mathcal{A}^*$ corrupting a corresponding set of Alice (Bob) instances such that $\mathcal{A}^*$ produces a transcript of messages and adversary view that is computationally indistinguishable from that produced by $\mathcal{A}$.

4. An OT Protocol without Zero-Knowledge Proofs

We arrive at our concurrent $\binom{2}{1}$-OT protocol through a series a transformations. We start by describing a non-interactive (stand-alone) $\binom{2}{1}$-OT protocol due to Bellare and Micali [5]. We transform it into an interactive $\binom{2}{1}$-OT protocol that uses witness-hiding proofs of knowledge, which we

[7]We will use perfect indistinguishability here to simplify our proofs, but we believe our results would generalize to computational indistinguishability.

Alice		**Bob**
(b_0, b_1)		(e)
Picks $\alpha_0, \alpha_1 \in_R \{0, \cdots, p-2\}$ Computes $a_0 = g^{\alpha_0}, a_1 = g^{\alpha_1}$, $d_0 = y_0^{\alpha_0}, d_1 = y_1^{\alpha_1}$ Picks $r_0, r_1 \in_R \{0,1\}^k$ s.t. $\langle d_0, r_0 \rangle = b_0, \langle d_1, r_1 \rangle = b_1$	$\xrightarrow{a_0, a_1, r_0, r_1}$	Computes $a_e^{x_e} = d_e$ $b_e = \langle d_e, r_e \rangle$

Figure 1. *The Bellare-Micali non-interactive OT protocol; Bob's public key is (y_0, y_1) and his secret key is (e, x_e)*

then (in Section 5) turn into a concurrent $\binom{2}{1}$-OT protocol with slightly non-constant number of rounds.

4.1. The Bellare-Micali $\binom{2}{1}$-OT protocol

Let prime p, generator g of Z_p^*, and element $m \in Z_p^*$ be known to all users; it is assumed that nobody knows the discrete log of m. Bob chooses his secret and public keys as follows: he picks $e \in \{0,1\}$ and $x_e \in \{0, \cdots, p-2\}$ at random, and sets $y_e = g^{x_e}$ and $y_{1-e} = \frac{m}{g^{x_e}}$. His public key is (y_0, y_1) and his secret key is (e, x_e). Anybody can check that Bob's public key is correctly formed, by checking that $y_0 y_1 = m$. Since the discrete log of m is not known, Bob should not know the discrete logs of both y_0 and y_1 (assuming finding discrete logs is infeasible); moreover, the public key does not reveal which of the two discrete logs Bob knows. Alice has a pair of bits (b_0, b_1). The protocol is shown in Figure 1. In the figure, $\langle x, y \rangle$ denotes the inner product mod 2 of the strings x and y, and $k = |p|$.

Bellare and Micali note that to prove security, a Bob must also provide a zero knowledge proof of knowledge of his secret key. However, no proof of security is given. Actually, it is not clear how to prove security in Beaver's security model, although including the zero-knowledge proof of knowledge seems to make their protocol "halfway" provably secure (that is, simulatable for the case when the adversary corrupts Bob). This proof would be based on the computational Diffie-Hellman problem (and uses the Goldreich-Levin theorem [34]).

4.2. The new stand-alone OT protocol

We now present a modification of the protocol above that achieves secure $\binom{2}{1}$-OT. Our protocol only uses a witness-hiding proof of knowledge (POK) [31] in place of the zero-knowledge POK from Bob to Alice that would be used in the protocol above. It also uses another witness-hiding POK from Alice to Bob after the first transfer in the protocol. The protocol is shown in Figure 3. In the figure, **POK**[$\cdot$] stands for a (witness-hiding) proof of knowledge protocol, namely, a so-called Σ-protocol [23] consisting of three moves: an initial message from the prover (in this case, Bob), a challenge, and a response. (Messages, of course, could be combined; we show them separately for clarity.) The proofs are of disjunctions and conjunctions of statements. Efficient proofs of monotone (boolean) composition of knowledge statements were given in [14, 32].

Before describing the protocol more thoroughly, we first explain a particular construction that our protocol uses, namely, proofs of knowledge of representations over multiple bases, and, in particular, with the extra condition that the representation over one of them is non-zero. (These representations are denoted REP$(y, (g_1, g_2, \cdots, g_k))$ and NZ-REP$_{g_i}(y, (g_1, g_2, \cdots, g_k))$, resp.—see Section 2.) This type of proof is needed for Alice to have a guarantee that Bob only knows the discrete log of one of the two public keys. To our knowledge, a proof of knowledge for this type of special representation is new and might be of more general interest.

Proof of knowledge of the special representation. For simplicity, we present here the POK of the particular representation that will be used in our OT protocol: over three bases, with one representation being non-zero; we defer the general cases (1 out of k, several bases out of k) to the full version of the paper. The protocol is shown in Figure 2. representation

Lemma 1 *The protocol of Figure 2 is a witness-indistinguishable proof of knowledge protocol for NZ-REP$_n(y, (g, m, n))$.*

Proof Completeness is straightforward. For soundness, using the knowledge extractor associated with the proof of knowledge, let (t, t', t'') be the representation extracted from the first part of the proof (i.e., $y = g^t m^{t'} n^{t''}$); (s, s', s'') the representation extracted from the second ($z = y^s g^{s'} m^{s''}$); u the discrete log extracted from the third ($w = n^u$); and (v, v') the representation extracted from

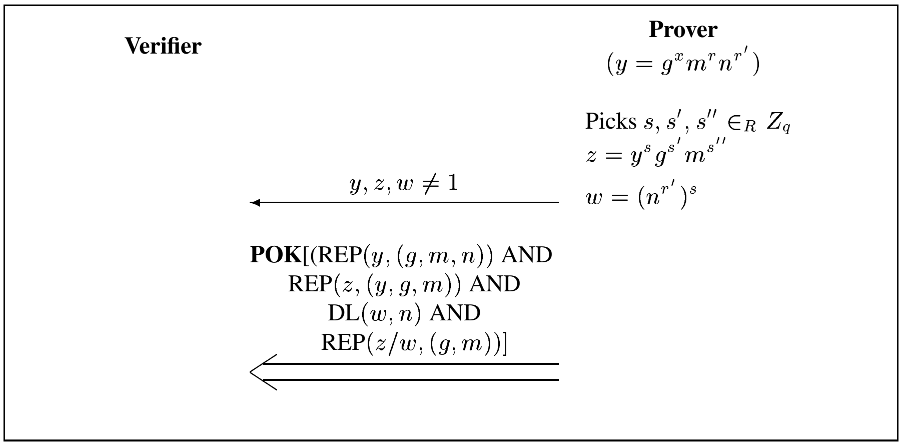

Figure 2. *Proof of knowledge of NZ-REP$_n(y,(g,m,n))$*

the fourth ($z/w = g^v m^{v'}$). If $t'' \neq 0$ then the assertion holds, so assume otherwise. From proofs 2–4 we obtain $z = y^s g^{s'} m^{s''} = g^v m^{v'} n^u$, which yields $y^s = g^{v-s'} m^{v'-s''} n^u$. There are two cases:

$s \neq 0$: Then $y = g^{\frac{v-s'}{s}} m^{\frac{v'-s''}{s}} n^{\frac{u}{s}}$. Since $w \neq 1$ and thus $u \neq 0$, this is a proper representation for y.

$s = 0$: Then $n = g^{\frac{s'-v}{u}} m^{\frac{s''-v'}{u}}$, and thus the verifier can compute a proper representation for y (say, $y = g^{t-\frac{s'-v}{u}} m^{t'-\frac{s''-v'}{u}} n$).

To prove witness indistinguishability, note that each proof of knowledge of a representation is witness indistinguishable, so even if the discrete logs of y, z, w, m and n are known (say c_1, c_2, c_3, a, and a' respectively), the verifier is only able to learn the following restrictions (all $\bmod q$):

$$\begin{aligned} c_1 &= x + ar + a'r', \\ c_2 &= c_1 s + s' + as'', \\ c_3 &= a'r's. \end{aligned}$$

This system of equations has a solution for any (x, r, r') satisfying $c_1 = x + ar + a'r'$. Note that no hardness assumptions are needed. $\square$

The protocol. The new (stand-alone) $\binom{2}{1}$-OT protocol is shown in Figure 3. The basic idea is the same as that in the Bellare-Micali protocol. Alice gives Bob a value n for which Bob does not know the discrete log (this n corresponds to the value m in the Bellare-Micali protocol), and forces Bob to commit to two values, one of them containing the value n. However, we want to avoid ZK protocols, so everything is done with representations. This disallows the use of the simple $(g^x, m/g^x)$ trick which forces Bob to use m in one of his two values, and thus Bob has to do a more complicated "NZ-REP" protocol, and include it in a more complicated boolean POK expression. On the Alice side, any reduction from Diffie-Hellman will require the value that the simulator sends for Alice in the first message to include a value for which the simulator does not know the discrete log. This is why we send two values (m, n), and simply prove knowledge of a representation of n over m and g. This allows the simulator to incorporate a value with an unknown discrete log into m.

Theorem 1 *Assuming the hardness of* DDH*, the protocol of Figure 3 is a secure stand-alone OT protocol.*

Proof [Sketch] Verification of the completeness requirement is straightforward. For simulatability, we will transform a real-world adversary $\mathcal{A}$ into an ideal-world adversary $\mathcal{A}^*$ that attacks the ideal system and outputs a transcript which is computationally indistinguishable from the one produced by $\mathcal{A}$ attacking the real system. (We do this essentially by constructing a simulator that simulates the user instances in the real system using corresponding user instances in an ideal system.)

Say $\mathcal{A}$ plays the role of real-Alice (Alice in the real system). Then $\mathcal{A}^*$ plays the role of ideal-Alice and simulates real-Bob for $\mathcal{A}$. $\mathcal{A}^*$ accepts m and n and then runs the knowledge extractor for the first POK. From this it receives REP$(n,(g,m))$, say $n = g^s m^{s'}$. Then $\mathcal{A}^*$ chooses $x, x' \in_R Z_q$ and sets $y_0 = g^x$ and $y_1 = nm^{-s'} g^{x'-s} = g^{x'}$. ¿From this point on, $\mathcal{A}^*$ follows the real-Bob protocol until the last step. When it receives a_0, d_0, a_1, d_1, it uses its knowledge of DL(y_0, g) and DL(y_1, g) to compute both b_0 and b_1. It sends (b_0, b_1) to the trusted host, and has ideal-Bob send its e value to the trusted host. (Note that $\mathcal{A}^*$ does not know the bit e at this point.) The trusted host sends ideal-Bob the requested bit b_e, and the pair (e, b_e) is output. This will be perfectly indistinguishable from what would be output in the real world.

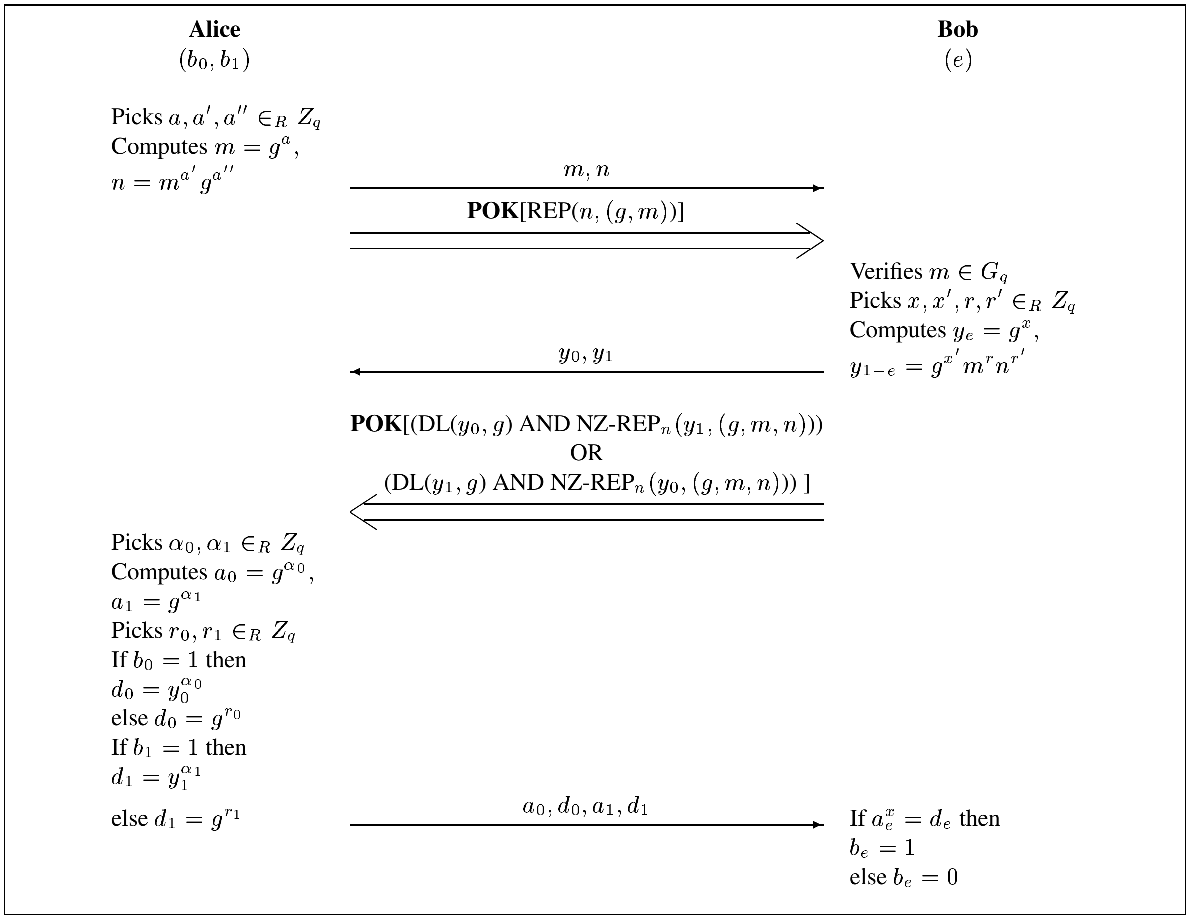

Figure 3. *A stand-alone $\binom{2}{1}$-OT protocol*

Now say $\mathcal{A}$ plays the role of real-Bob. Then $\mathcal{A}^*$ plays the role of ideal-Bob and simulates real-Alice. $\mathcal{A}^*$ follows the protocol for real-Alice, except that using the extractor associated with the second proof of knowledge, it extracts the discrete log x of one of (y_0, y_1) from $\mathcal{A}$ (who is playing real-Bob), and a representation (x', r, r') of the other one. If the extraction fails, or if the extractor extracts the discrete logs and representations of both, then $\mathcal{A}^*$ aborts. (Note that the extraction fails only with probability $\frac{1}{q}$, and the other case will happen only with negligible probability because $\mathcal{A}$ would essentially be computing discrete logs, as shown below.) Say $\mathcal{A}^*$ extracts the discrete log of y_e for $e \in \{0, 1\}$. Then $\mathcal{A}^*$ knows that e is the index of the bit that $\mathcal{A}$ wants. $\mathcal{A}^*$ has ideal-Alice send her bits b_0 and b_1 to the trusted host, and then sends e to the trusted host as ideal-Bob. (Note that $\mathcal{A}^*$ doesn't know the bits b_0 and b_1 at this point.) $\mathcal{A}^*$ receives b_e back from the trusted host, so now $\mathcal{A}^*$ knows b_e, but not b_{1-e}. $\mathcal{A}^*$ then computes a_e and d_e as in the real protocol, but chooses random values for a_{1-e} and d_{1-e}.

If there is a distinguisher D that can distinguish the transcript of $\mathcal{A}$ from the transcript of $\mathcal{A}^*$, then we can create a distinguisher D' for DDH as follows. Take a DDH challenge: (X, Y, Z) and set $m = X$. Run the protocol as real-Alice against an adversary $\mathcal{A}$ who plays the role of real-Bob. Also run a (fresh) copy of $\mathcal{E}$ to generate inputs to the protocol. Now extract the discrete log x of y_e and representation (x', r, r') of y_{1-e} as in the simulation above, where e is the bit that $\mathcal{A}$ wants. If the extractor extracts the discrete logs and representations of both, then D' simply computes the discrete log of X and computes the correct answer to the DDH challenge directly. If the extraction fails, or extracts both discrete logs and representations of both y_0 and y_1 but is not able to compute the discrete log of X, D' answers "Random" and stops. (This happens with probability at most $\frac{1}{q} + \frac{1}{q} = \frac{2}{q}$.)

Now obtain input bits b_0 and b_1 from the copy of $\mathcal{E}$. In the last step, choose a random r', set $a_{1-e} = Y$ and either set $d_{1-e} = Z^{a'r'+r} Y^{a''r'+x'}$ if $b_{1-e} = 1$, or set d_{1-e} randomly if $b_{1-e} = 0$. Now run D on the transcript. If

D answers "real protocol," then answer "True DH." If D answers "simulated protocol," then answer "Random."

Let D-real be the event that D answers "real" and D-sim the event that D answers "simulated," and let $r = \Pr(D\text{-real}|\text{"real"})$ and let $r' = \Pr(D\text{-real}|\text{"sim"})$. Since D distinguishes with non-negligible probability, we have that for some non-negligible ϵ:

$$\begin{aligned}\epsilon &\leq \Pr(D\text{-real}|\text{"real"}) - \Pr(D\text{-real}|\text{"sim"}) \\ &= r - r'.\end{aligned}$$

Let E_1 be the event that the extraction fails, and let E_2 be the event that the extraction succeeds but we cannot compute e, and furthermore cannot compute $\mathrm{DL}(h_u, g)$. Let $w = \Pr(E_1)$ and let $w' = \Pr(E_2)$. Let E_3 be the event that the extraction succeeds and allows the simulator to compute $\mathrm{DL}(X, g)$ directly. Let $w'' = \Pr(E_3)$. Note that w, w', and w'' are the same whether $Z = \mathrm{DH}(X, Y)$ or $Z \neq \mathrm{DH}(X, Y)$, and that $w + w' \leq \frac{2}{q}$. Also note that when $Z = \mathrm{DH}(X, Y)$, Alice's responses are exactly as in the real protocol, and when $Z \neq \mathrm{DH}(X, Y)$, Alice's responses are exactly as in the simulated protocol. Therefore,

$$\begin{aligned}&\Pr(D' \text{ is correct}) \\ &= \Pr(D\text{-real}|Z = \mathrm{DH}(X,Y)) \Pr(Z = \mathrm{DH}(X,Y)) + \\ &\quad \Pr(D\text{-sim}|Z \neq \mathrm{DH}(X,Y)) \Pr(Z \neq \mathrm{DH}(X,Y)) + \\ &\quad \Pr(E_1|Z \neq \mathrm{DH}(X,Y)) \Pr(Z \neq \mathrm{DH}(X,Y)) + \\ &\quad \Pr(E_2|Z \neq \mathrm{DH}(X,Y)) \Pr(Z \neq \mathrm{DH}(X,Y)) + \\ &\quad \Pr(E_3) \\ &\geq (r - w - w' - w'') \left(\frac{1}{2}\right) + \\ &\quad (1 - r' - w - w' - w'') \left(\frac{1}{2}\right) + \\ &\quad (w + w') \left(\frac{1}{2}\right) + w'' \\ &= \frac{1}{2} + \frac{r - r' - w - w'}{2} \\ &\geq \frac{1}{2} + \frac{\epsilon}{2} - \frac{1}{q},\end{aligned}$$

which is non-negligibly more than 1/2. Thus, if there is a distinguisher that can distinguish the transcript from the real world from that of the ideal world, then the DDH distinguisher constructed above would have a non-negligible advantage over a random choice.

This completes the proof of the theorem. □

5. The Concurrent OT Protocol

The protocol for concurrent $\binom{2}{1}$-OT is obtained from the protocol of Figure 3 by running *sequentially* M proofs of knowledge of the same statement from Bob to Alice instead of just one, where M is any non-constant (i.e., positive, increasing) function of the security parameter. (Note that the POK from Alice to Bob, i.e., the first POK in the protocol, is still only run once.) For a given user instance and for $s \in \{1, \ldots, M\}$, we use $\mathbf{POK}_s$ to denote the sth POK from Bob to Alice in the sequence for that user instance. For efficiency, the initial message in each POK can be combined with the final message from the preceding POK. To simplify exposition, however, we will assume that the messages remain separate.

The reason we run the sequential proofs of knowledge is that the extractor for the proof of knowledge is not able to run exactly the same execution as the normal simulator, at least not without having to deal with possible nested rewinding and interleaving of proofs of knowledge. In particular, the normal simulator will always transmit random noise for the bit that cannot be read by Bob (because this bit is unknown to the normal simulator), while the extractor will not know the index of the bit that will be read by Bob, and thus must transmit both bits (drawn from a simulated distribution) in the correct way. Given that the views seen by the adversary running against the extractor and the normal simulator are computationally indistinguishable, but nevertheless different, the basic argument that the extractor finishes in polynomial time does not hold. To get around this problem, we make sure the extractor for a given POK fails with polynomially small probability, and then after a non-constant number of independent attempts fails with only negligible probability.

5.1. Proof of security

We prove security by specifying a simulator that uses an ideal system to simulate adversary operations in a real system, so that the transcript of an adversary attacking the simulator is computationally indistinguishable from the transcript of an adversary attacking the real system. We will assume the environment satisfies conditional simulatability.

Let T be the number of operations the adversary performs in the real system. T must be polynomial in the security parameter k, and without loss of generality we will assume $T \geq k$.

The Alice simulator. Here we describe the actions of the simulator for each possible operation of a "Bob-controlling" adversary in the real protocol. We name the actions of the real user instances as follows:

Alice0 Alice instance action to start the protocol (i.e., an Alice0 action for a real Alice instance is sending a message (m, n), and sending the first message in the first POK).

Alice-challenge Alice instance action to respond to the challenge in the first POK.

Alice$(s, 1)$ Alice instance action on receiving the initial message in $\mathbf{POK}_s$.
Alice$(s, 2)$ Alice instance action on receiving the response message in $\mathbf{POK}_s$.

The simulator performs actions as follows:

1. Alice0, Alice-challenge, Alice$(s, 2)$ for $1 \leq s \leq M-1$: The simulator behaves just as a real Alice instance would.

2. Alice$(s, 1)$ for $1 \leq s \leq M$: The simulator starts T^2 separate *look-ahead* simulations starting at the current operation and running until the adversary stops. (Recall that the adversary performs at most T operations total.) In each one, the simulator does not use the ideal system, but uses the conditional simulatability property of the environment to generate inputs for user instances from the correct input distribution, and behaves as in the real protocol. In each look-ahead, a random challenge is used for this proof of knowledge (i.e., $\mathbf{POK}_s$ in this Alice instance), and we say that the look-ahead succeeds if a valid response for that challenge is received. Otherwise, we say the look-ahead fails.

   After all look-ahead simulations are finished, the simulator proceeds as a real Alice instance would (i.e., the simulator generates and sends a challenge for the POK).

   Question: Why can't we use a simple extractor (that succeeds with all but negligible probability) for these POKs instead of the T^2 look-ahead simulations?

   Answer: A simple extractor would rewind after obtaining the last message of the POK from the adversary. But how do we perform the rewind phase with the exact same distribution as the normal simulation? This seems to require us to know the e values used by Bob instances in concurrent executions, which leads to more rewinding in other POKs and thus the same type of concurrency problems encountered in concurrent ZK proofs. On the other hand, if the distributions are not exactly equal in the normal simulation and the rewind phase, we do not know how to construct an expected polynomial time extractor that succeeds (either because it extracted the desired knowledge, or because the normal simulation never received an answer in the POK) with all but negligible probability, even though the difference in success probabilities in the two types of simulation may be negligible. For instance, say the normal simulation succeeds with probability $1/\sqrt{q}$ and the look-ahead succeeds with probability $1/q$. The difference between these probabilities is negligible, but the expected simulation time would be about $\sqrt{q}$, which is super-polynomial.

   Since we cannot obtain negligible probability of success in the extractor, we run T^2 look-aheads (which are similar to rewind phases, but run before the normal simulation instead of after) to make the probability of failure polynomially small, and then run a non-constant number of POKs to force the probability of all POKs failing to be negligible.

3. Alice$(M, 2)$: If all look-aheads for all POKs have failed, or if all the look-aheads that succeed use exactly the same challenge as the simulator used in the normal (non-look-ahead) simulation, the simulator aborts the simulation. Otherwise, the simulator extracts the values x and (x', r, r') from two responses to two distinct challenges in one of the POKs. The simulator can then determine the value e used by the adversary. (If it cannot, because it obtains discrete logs and representations for both y_0 and y_1, the simulator simply aborts.) The simulator sends that value e to the trusted host in the ideal system, and has the ideal-Alice instance send her bits b_0 and b_1. (Note that the simulator does not know b_0 and b_1 at this point.) The simulator then receives the bit b_e.

   Now the simulator generates a_e and d_e as in the real protocol, and draws a_{1-e} and d_{1-e} uniformly from G_q. The simulator sends these values to Bob.

The Bob simulator. The actions of the simulator against an Alice-controlling adversary in the real protocol:
Bob0 Bob instance action on receiving the first message from Alice.
Bob1 Bob instance action on receiving the first message of the first POK.
Bob$(s, 1)$ Bob instance action to start $\mathbf{POK}_s$.
Bob$(s, 2)$ Bob instance action to respond to the challenge in $\mathbf{POK}_s$.
Bob-final Bob instance action to respond to the last message from Alice.

The simulator performs the actions as follows:

1. Bob0, Bob$(s, 1)$ for $2 \leq s \leq M$, and Bob$(s, 2)$ for $1 \leq s \leq M$: The simulator behaves just as a real Bob instance would.

2. Bob1: The simulator saves the state of the execution for a later rewind to this point. Then the simulator generates and sends a challenge, as a real Bob instance would.

3. Bob$(1, 1)$: If the POK succeeds, Bob rewinds to the Bob1 step, and uses the knowledge extractor to ex-

tract REP$(n, (g, m))$. After rewinding, the simulator performs a type of look-ahead simulation as the Alice simulator did above that uses no interaction with the ideal world and no further rewinding. Essentially, the look-ahead simulation chooses the e value for each Bob instance by using the conditional simulatability property of the environment. The reason that this works is that

3.1 Bob instances that have gotten to step Bob$(1, 1)$ have already computed y_0 and y_1 values such that the simulator knows the discrete log of both (see below), and can thus compute b_e sent from Alice for either $e = 0$ or $e = 1$, and

3.2 Bob instances that have not gotten to step Bob$(1, 1)$ are free to set their y_0 and y_1 according to the e chosen by the simulator.

Thus the look-ahead simulation performed during rewinding is perfectly indistinguishable from the normal simulation, and thus the extraction can succeed in expected polynomial time (as in a stand-alone proof-of-knowledge).

After extracting REP$(n, (g, m))$, the simulator sets $y_0 = g^x$ and $y_1 = nm^{-s'} g^{x'-s} = g^{x'}$, i.e., it knows the discrete log of both.

4. Bob-final: Using its knowledge of the discrete logs of both y_0 and y_1, the simulator determines the bits b_0 and b_1 sent by Alice (controlled by the adversary), and sends them to the trusted host as the ideal-Alice instance. Then the simulator has the ideal-Bob instance send its bit e to the trusted host. (Note that the adversary does not know e at this point.) The ideal-Bob instance outputs (e, b_e).

This completes the description of the Bob simulator.

Theorem 2 *Assuming the hardness of* DDH*, the protocol described in Section 5 is a secure Concurrent OT protocol, assuming conditional simulatability of the environment.*

Proof [Sketch] First consider the case of an Alice-controlling adversary $\mathcal{A}$. Let $\mathcal{A}^*$ be defined as $\mathcal{A}$ running on the simulator described above. (Thus $\mathcal{A}^*$ is an adversary for the ideal system.) Since the simulation of Bob-instances is perfectly indistinguishable from a set of real-Bob instances, if $\mathcal{A}$ could break the real system, then $\mathcal{A}^*$ could break the ideal system, which is impossible.

Now consider the case of a Bob-controlling $\mathcal{A}$. Let $\mathcal{A}^*$ be defined as $\mathcal{A}$ running on the simulator described above. (Again, $\mathcal{A}^*$ is an adversary for the ideal system.) If there is a distinguisher D that can distinguish the transcript of $\mathcal{A}$ (running on the real system) from the transcript of $\mathcal{A}^*$, then we can create a distinguisher D' for DDH as follows. Take a DDH challenge: $v = (X, Y, Z)$. Whenever necessary, generate a new DDH instance (h_u, h'_u, h''_u) which is a DH-triple if and only if the challenge is. (This can be done as described in [49].) Run $\mathcal{A}$ against the simulator described above with the following modifications:

- When simulating Alice0, set $m = h_u$ (for a new u).
- When simulating Alice$(M, 2)$
  1. Choose b_0, b_1 from the correct input distribution (conditioned on all previous inputs).
  2. If $b_{1-e} = 1$, set $a_{1-e} = h'_u$ and $d_{1-e} = (h''_u)^{a'r'+r}(h'_u)^{a''r'+x'}$. Otherwise set a_{1-e} and d_{1-e} randomly. (This makes the protocol equivalent to the real protocol if and only if the DDH challenge is a DH-triple.)

If D answers "real protocol," then guess "DH-triple;" if D answers "simulated protocol," then guess "random triple;" if the simulator for D' aborts because the extraction failed, or because e could not be determined and furthermore DL(h_u, g) could not be determined, then guess "DH-triple;" if the simulator for D' aborts because e could not be determined but DL(h_u, g) could be (say it is γ), then if $(h'_u)^\gamma = h''_u$, guess "DH-triple" else guess "random triple."

Let D-real be the event that D answers "real" and D-sim the event that D answers "simulated," and let $r = \Pr(D\text{-real}|\text{"real"})$ and $r' = \Pr(D\text{-real}|\text{"sim"})$. Since D distinguishes with non-negligible probability, we have that for some non-negligible ϵ:

$$\begin{aligned} \epsilon &\leq \Pr(D\text{-real}|\text{"real"}) - \Pr(D\text{-real}|\text{"sim"}) \\ &= r - r'. \end{aligned}$$

Let E_1 be the event that an extraction fails, and let E_2 be the event that an extraction succeeds but we cannot compute e, and furthermore cannot compute DL(h_u, g). Let E_3 be the event that an extraction succeeds and allows the simulator to compute DL(h_u, g) directly. For $i = 1, 2, 3$, let $w_i = \Pr(E_i|Z = \mathrm{DH}(X, Y))$ and let $w'_i = \Pr(E_i|Z \neq \mathrm{DH}(X, Y))$. (In contrast to the proof for the standalone protocol, w_i may not be equal to w'_i.) Note that $w_2, w'_2 \leq \frac{T}{q}$. Also note that when $Z = \mathrm{DH}(X, Y)$, Alice's responses are exactly as in the real protocol, and when $Z \neq \mathrm{DH}(X, Y)$, Alice's responses are exactly as in the simulated protocol. Therefore,

$$\begin{aligned} &\Pr(D' \text{ is correct}) \\ &\quad= \Pr(D\text{-real}|Z = \mathrm{DH}(X, Y)) \Pr(Z = \mathrm{DH}(X, Y)) + \\ &\qquad \Pr(D\text{-sim}|Z \neq \mathrm{DH}(X, Y)) \Pr(Z \neq \mathrm{DH}(X, Y)) + \\ &\qquad \Pr(E_1|Z = \mathrm{DH}(X, Y)) \Pr(Z = \mathrm{DH}(X, Y)) + \\ &\qquad \Pr(E_2|Z = \mathrm{DH}(X, Y)) \Pr(Z = \mathrm{DH}(X, Y)) + \\ &\qquad \Pr(E_3) \\ &\quad\geq (r - w_1 - w_2 - w_3)\left(\frac{1}{2}\right) + \end{aligned}$$

$$
\begin{aligned}
& (1 - r' - w_1' - w_2' - w_3')\left(\frac{1}{2}\right) + \\
& (w_1 + w_2)\left(\frac{1}{2}\right) + \frac{w_3 + w_3'}{2} \\
= \; & \frac{1}{2} + \frac{r - r' - w_1' - w_2'}{2} \\
\geq \; & \frac{1}{2} + \frac{\epsilon}{2} - \frac{w_1'}{2} - \frac{T}{q},
\end{aligned}
$$

which is non-negligibly more than 1/2, as long as w_1' is negligible.

Claim 1 *w_1' is negligible.*

Proof w_1' is the probability that in one of the at most T Alice-Bob instances, every set of look-aheads in the M proofs fail. We will show that the probability of any set of look-aheads failing for a given POK is $1/T^2$ (independently of any other set of look-aheads), and thus the probability of the simulator aborting is T/T^{2M}, which is negligible.

First, for a given POK in the simulation, note that the probability that $\mathcal{A}$ gives a response in the simulation is the same as the probability that $\mathcal{A}$ gives a response in the look-ahead, since they both have the same distribution (because $Z \neq \mathrm{DH}(X,Y)$). Let p be this probability.

Then the probability of getting a response in the simulation, when all of T^2 look-aheads fail is at most $p(1-p)^{T^2} \leq 1/T^2$. This completes the proof of the claim. □

This completes the proof of the theorem. □

6. Summary and Open Problems

In this paper we introduced a model for proving security of concurrent OT protocols, and presented a protocol that is proven secure in this model based on the DDH problem. The protocol is efficient, requiring only a slightly non-constant number of rounds. One immediate problem is to design a concurrent OT protocol with a constant number of rounds.

One could also consider a stronger adversarial model for concurrent OT in which an adversary may simultaneously corrupt parties playing Alice and Bob roles, and may modify messages sent between honest parties. This model could also be considered for concurrent ZK, but a more basic model is generally used to highlight the fundamental problems underlying concurrency in ZK protocols. We have chosen to use a more basic model for concurrent OT for the same reason—to highlight the fundamental problems underlying concurrency in OT protocols. The stronger adversarial model (for either concurrent OT or concurrent ZK) generally requires the use of non-malleable primitives [26] to prove security, and thus causes protocols to be more complex and less efficient.

Acknowledgements We thank Yehuda Lindell for pointing out that in the original version of this paper (in FOCS 2000), we incorrectly stated that our protocol was proven secure with respect to any distribution of inputs, when in fact we only prove it to be secure with respect to inputs that satisfy conditional simulatability.

References

[1] D. Beaver. *Security, Fault Tolerance, and Communication Complexity*. PhD thesis, Harvard University, 1990.

[2] D. Beaver. How to break a "secure" oblivious transfer protocol. In *Advances in Cryptology—EUROCRYPT 92*, volume 658 of *Lecture Notes in Computer Science*, pages 285–296. Springer-Verlag, 24–28 May 1992.

[3] D. Beaver. Equivocable oblivious transfer. In *Advances in Cryptology—EUROCRYPT 96*, volume 1070 of *Lecture Notes in Computer Science*, pages 119–130. Springer-Verlag, 12–16 May 1996.

[4] M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In STOC'98 [50], pages 419–428.

[5] M. Bellare and S. Micali. Non-interactive oblivious transfer and applications. In CRYPTO'89 [20], pages 547–557.

[6] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Advances in Cryptology—CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer-Verlag, 22–26 Aug. 1993.

[7] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, and M. Yung. Systematic design of two-party authentication protocols. In *Advances in Cryptology—CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 44–61. Springer-Verlag, 1992, 11–15 Aug. 1991.

[8] M. Boyarsky. Public-key cryptography and password protocols: The multi-user case. In CCS'99 [13], pages 63–72.

[9] G. Brassard, C. Crepeau, and J. Robert. Information theoretic reductions among disclosure problems. In *27th Annual Symposium on Foundations of Computer Science*, pages 168–173, Toronto, Ontario, Canada, 27–29 Oct. 1986. IEEE.

[10] G. Brassard, C. Crépeau, and J.-M. Robert. All-or-nothing disclosure of secrets. In *Advances in Cryptology—CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 234–238. Springer-Verlag, 1987, 11–15 Aug. 1986.

[11] C. Cachin. On the foundations of oblivious transfer. In *Advances in Cryptology—EUROCRYPT 98*, volume 1403 of *Lecture Notes in Computer Science*, pages 361–374. Springer-Verlag, 1998.

[12] R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali. Resettable zero-knowledge. STOC 2000.

[13] *Proceedings of the Sixth Annual ACM Conference on Computer and Communications Security*, 1999.

[14] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology—CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer-Verlag, 21–25 Aug. 1994.

[15] C. Crépeau. Equivalence between two flavours of oblivious transfers. In *Advances in Cryptology—CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 350–354. Springer-Verlag, 1988, 16–20 Aug. 1987.

[16] C. Crépeau. Verifiable disclosure of secrets and applications (abstract). In *Advances in Cryptology—EUROCRYPT 89*, volume 434 of *Lecture Notes in Computer Science*, pages 150–154. Springer-Verlag, 1990, 10–13 Apr. 1989.

[17] C. Crépeau, J. van de Graaf, and A. Tapp. Committed oblivious transfer and private multi-party computation. In *Advances in Cryptology—CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 110–123. Springer-Verlag, 27–31 Aug. 1995.

[18] G. D. Crescenzo and R. Ostrovsky. On concurrent zero-knowledge with pre-processing. In Wiener [51], pages 485–502.

[19] G. D. Crescenzo, R. Ostrovsky, and S. Rajagopalan. Conditional oblivious transfer and timed-release encryption. In EuroCrypt'99 [29], pages 74–89.

[20] *Advances in Cryptology—CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990, 20–24 Aug. 1989.

[21] I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *Advances in Cryptology—EUROCRYPT '2000*, Lecture Notes in Computer Science, pages 418–430. Springer-Verlag, 14–18 May 2000.

[22] I. Damgård, J. Kilian, and L. Salvail. On the (im)possibility of basing oblivious transfer and bit commitment on weakened security assumptions. In EuroCrypt'99 [29], pages 56–73.

[23] I. B. Damgård. On the existence of bit commitment schemes and zero-knowledge proofs. In CRYPTO'89 [20], pages 17–27.

[24] W. Diffie, P. van Oorschot, and M. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2:107–125, 1992.

[25] Y. Dodis and S. Micali. Lower bounds for oblivious transfer reductions. In EuroCrypt'99 [29], pages 42–55.

[26] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography (extended abstract). In *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, pages 542–552, New Orleans, Louisiana, 6–8 May 1991.

[27] C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. In STOC'98 [50], pages 409–418.

[28] C. Dwork and A. Sahai. Concurrent zero-knowledge: Reducing the need for timing constraints. In H. Krawczyk, editor, *Advances in Cryptology—CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 442–457. Springer-Verlag, 17–21 Aug. 1998.

[29] *Advances in Cryptology—EUROCRYPT 99*, volume 1592 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

[30] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts (extended abstract). In *Advances in Cryptology: Proceedings of Crypto 82*, pages 205–210. Plenum Press, New York and London, 1983, 23–25 Aug. 1982.

[31] U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 416–426, Baltimore, Maryland, 14–16 May 1990.

[32] J. Garay, M. Jakobsson, and P. MacKenzie. Abuse-free optimistic contract signing. In Wiener [51], pages 449–466.

[33] O. Goldreich. Secure multi-party computation. manuscript. Available from `http://www.wisdom.weizmann.ac.il/~oded/PS/prot.ps`.

[34] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 25–32, Seattle, Washington, 15–17 May 1989.

[35] S. Halevi and H. Krawczyk. Public-key cryptography and password protocols. In *Proceedings of the Fifth Annual ACM Conference on Computer and Communications Security*, pages 122–131, 1998.

[36] D. Harkins and D. Carrel. The internet key exchange. *RFC 2409*, 1998.

[37] L. Harn and H. Lin. Noninteractive oblivious transfer. *Electronics Letters*, 26(10):635–636, 1990.

[38] J. Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 20–31, Chicago, Illinois, 2–4 May 1988.

[39] J. Kilian, S. Micali, and R. Ostrovsky. Minimum resource zero-knowledge proofs (extended abstract). In CRYPTO'89 [20], pages 545–546.

[40] J. Kilian and E. Petrank. Concurrent zero-knowledge in poly-logarithmic rounds (extended abstract). Available from the Cryptology ePrint Archive, `http://eprint.iacr.org/2000/013`.

[41] J. Kilian, E. Petrank, and C. Rackoff. Lower bounds for zero knowledge on the internet. In *39th Annual Symposium on Foundations of Computer Science*, pages 484–492. IEEE, Nov. 1998.

[42] S. Lucks. Open key exchange: How to defeat dictionary attacks without encrypting public keys. In *Proceedings of the Workshop on Security Protocols*, 1997.

[43] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, pages 245–254, Atlanta, Georgia, 1–4 May 1999.

[44] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[45] M. O. Rabin. How to exchange secrets by oblivious transfer. Technical Report Tech. Memo TR-81, Aiken Computation Laboratory, Harvard University, 1981.

[46] R. Richardson and J. Kilian. On the concurrent composition of zero-knowledge proofs. In EuroCrypt'99 [29], pages 415–431.

[47] A. Rosen. A note on the round complexity of concurrent zero-knowledge. In M. Bellare, editor, *Advances in Cryptology—CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 451–468. Springer-Verlag, 20–24 Aug. 2000.

[48] A. D. Santis and G. Persiano. Public-randomness in public-key cryptography (extended abstract). In *Advances in Cryptology—EUROCRYPT 90*, volume 473 of *Lecture Notes in Computer Science*, pages 46–62. Springer-Verlag, 1991, 21–24 May 1990.

[49] V. Shoup. On formal models for secure key exchange. In CCS'99 [13], invited talk. Available from `http://www.shoup.net/papers/skey.ps.Z`.

[50] *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, Dallas, Texas, 23–26 May 1998.

[51] M. Wiener, editor. *Advances in Cryptology—CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*. Springer-Verlag, 15–19 Aug. 1999.