6.892 Computing on Encrypted Data October 28, 2013

Lecture 7

Lecturer: Vinod Vaikuntanathan *Scribe: Prashant Vasudevan*

1 Garbled Circuits

Picking up from the previous lecture, we start by defining a garbling scheme for circuits, formalised a suprisingly short time ago in [BHR12].

Definition 1. *Garbling Scheme*
A garbling scheme is a tuple of three algorithms (GarbleCkt, GarbleInp, GarbleEval).

- GarbleCkt *takes as input the description of a circuit C (on say n inputs, with 1 output) and security parameter 1^λ and outputs (GC, sk), where GC is the garbled circuit and sk is a key that may later be used to garble an input.*
- GarbleInp *takes as input the security parameter 1^λ, a key sk (that is supposed to have been previously generated by* GarbleCkt*) and an n-bit string x and outputs gx, which is the garbled input corresponding to x under sk.*
- GarbleEval *takes as input a garbled circuit GC and a garbled input gx and, if both are associated with the same key sk, computes $C(x)$, where GC is a garbling of C and gx of x.*

We require the following of any reasonable Garbling Scheme:

- *Correctness: $\forall (GC, gx)$ such that $(GC, sk) \leftarrow$ GarbleCkt(C) and $gx \leftarrow$ GarbleInp(x, sk), we need* GarbleEval$(GC, gx) = C(x)$
- *Efficiency: The running time of* GarbleInp *must be $poly(\lambda, |x|)$.*
- *Security: $\exists PPT$ simulator SIM such that $\forall (C, x) : SIM(C(x), 1^{|C|}, 1^{|x|}) \approx (GC, gx)$.*

The security condition goes to say that the garbled circuit and input reveal almost nothing more than the evaluation of the circuit on that input.

1.1 A scheme for inner products

Consider the following example of garbling for this family of functions that compute inner products: $\{f_y : \{0,1\}^{|y|} \to \{0,1\} : f_y(x) =< x, y >\}$.

- $GarbleCkt(C)$: $GC = \perp$, $sk = \begin{pmatrix} r_1 & r_2 & \cdots & r_n \\ r_1 \oplus y_1 & r_2 \oplus y_2 & \cdots & r_n \oplus y_n \end{pmatrix}$, where r_i's are random bits subject to $\bigoplus_i r_i = 0$.
- $GarbleInp(sk, x) : gx$ is computed as $gx_i = sk_{x_i,i}$ ($gx_i = r_i$ if $x_i = 0$ and $gx_i = r_i \oplus y_i$ if $x_i = 1$)
- $GarbleEval(GC, gx) = \bigoplus_i gx_i$

- This is correct because $x \wedge y$ is 0 if $x = 0$ and y when $x = 1$, and hence gx_i is always $r_i \oplus (x_i \wedge y_i)$. So we have:

$$\begin{aligned}\bigoplus_i gx_i &= \bigoplus_i (r_i \oplus (x_i \wedge y_i)) \\ &= \bigoplus_i r_i \oplus \bigoplus_i (x_i \wedge y_i) \\ &= 0 \oplus \bigoplus_i (x_i \wedge y_i) =< x, y >\end{aligned}$$

- It is also efficient because the work done by *GarbleInp* is only the selection of what to set gx_i as, which is linear in $|x|$.
- That it reveals nothing about x may be proven using the Leftover Hash Lemma.
  - Define a family of hash functions $\mathcal{H} = \{h_x\}_{x \in \{0,1\}^n}$ from $\{0,1\}^{2n} \to \{0,1\}^n$ such that $h_x(y, r)$ is computed similar to the garbled input in the above scheme.
  - $\mathcal{H}$ is pair-wise independent and (y, r) has $2n$ bits of entropy while the range of h_x is n-bit strings.
  - The Leftover Hash Lemma tells us that the distribution of $(x, h_x(y, r))$ is statistically close to (x, U) (where U is a uniform random variable).
  - Hence $h_x(y, r)$ (and so $\mathsf{GarbleInp}(x)$) reveals no information about x.

1.2 Relations to Functional encryption

We now look at how garbling circuits could give us (some form of) functional encryption. We start by reviewing the components of an FE scheme.

- $Setup \to (mpk, msk)$ (These are the master public and secret keys.)
- $KeyGen(msk, C) \to sk_C$ (This is the secret key that may be used to compute the circuit C on an encrypted input.)
- $Enc(mpk, x) \to CT$
- $Dec(sk_C, CT) \to C(x)$ when CT is a valid encryption of x.

Consider a garbling scheme that is such that the secret key generated by $\mathsf{GarbleCkt}$ is of the form $\begin{pmatrix} L_{10} & \cdots & L_{n0} \\ L_{11} & \cdots & L_{n1} \end{pmatrix}$ for some L_{ib}'s and $\mathsf{GarbleInp}(x)$ is $(L_{1x_1}, \cdots, L_{nx_n})$. The scheme for inner products presented above is an example of such a scheme.

Garbling schemes of this form may be used, along with a public key encryption scheme, to realise a modified notion of functional encryption as described below.

- *Setup*:
  - Obtain $2n$ key pairs $(pk_{ib}, sk_{ib})_{i \in [n], b \in \{0,1\}}$ from the encryption scheme.
  - Set $mpk = \begin{pmatrix} pk_{10} & \cdots & pk_{n0} \\ pk_{11} & \cdots & pk_{n1} \end{pmatrix}$, and $msk = \begin{pmatrix} sk_{10} & \cdots & sk_{n0} \\ sk_{11} & \cdots & sk_{n1} \end{pmatrix}$.
- $KeyGen(msk, x) : sk_x = (sk_{1x_1}, \cdots, sk_{nx_n})$ (Similar to the garbled input in the inner product scheme.)
- $Enc(mpk, C)$:
  - Obtain (GC, sk) from $\mathsf{GarbleCkt}$ of the garbling scheme.

- sk is of the form $\begin{pmatrix} L_{10} & \cdots & L_{n0} \\ L_{11} & \cdots & L_{n1} \end{pmatrix}$
- $CT_C = (GC, \{Enc_{pk_{ib}}(L_{ib})\}_{(i,b)})$

- $Dec(sk_x, CT_C)$:
  - Given sk_x and CT_C, it is possible to obtain $\mathsf{GarbleInp}(x) = (L_{1x_1}, \cdots, L_{nx_n})$ by using the keys in sk_x to decrypt the appropriate encryptions of the L_{ib}'s in CT_C.
  - The evaluation algorithm of the garbling scheme may then be used to find $C(x)$ given GC and $\mathsf{GarbleInp}(x)$.

The above realisation has the following shortcomings:

- Garbling schemes are single-use objects. They do not guarantee anything about security if a given garbled circuit is evaluated on more than one input.
- As the encryption algorithm involves encrypting the circuit and not the input, the size of the ciphertext produced grows with the size of the computation that is to be performed rather than with the size of the input.

1.3 Yao's Garbled Circuits

Before we look into resolving these issues, we present a construction of a garbling scheme due to Yao (in [Yao82]).

Without loss of generality, we assume that all circuits (of any given size) in the class of circuits we wish to garble have the same topology. That is, the underlying directed graph of each circuit is the same, and they differ only in the identities of the gates at the vertices of this graph. This may be ensured by adding dummy gates to circuits that have differing topologies.

We make use of a *checkable secret-key encryption scheme*. This is a secret-key encryption scheme with an additional algorithm check with the following properties:

- $\mathsf{check}(sk, CT) = \begin{cases} 1 & \text{if } \mathsf{Enc}_{sk}(m) = CT \text{ for some m} \\ 0 & \text{otherwise} \end{cases}$
- $\forall sk : Pr\left[A(1^\lambda) = CT' \mid \mathsf{check}(sk, CT) = 1\right] = negl(\lambda)$
  - That is, it is not possible to produce a valid ciphertext under a key without knowledge of the key.

The garbling scheme for such a class of circuits, with security parameter λ, is as follows:

- $\mathsf{GarbleCkt}(C)$:
  - For each wire w in the circuit, choose at random a pair of labels $\begin{pmatrix} L_{w0} \\ L_{w1} \end{pmatrix}$, $L_{wb} \in \{0,1\}^\lambda$. We shall use these labels to indicate the value carried by this wire.
  - The secret key is the set of pairs of labels corresponding to the input wires. That is,

$$sk = \begin{pmatrix} L_{10} & \cdots & L_{n0} \\ L_{11} & \cdots & L_{n1} \end{pmatrix}$$

  - For each gate g that has input wires u and v and output wire w, we compute the following 'table' using the encryption scheme:

$$\boxed{\begin{array}{l} Enc_{L_{u0}}(Enc_{L_{v0}}(L_{wg(0,0)})) \\ Enc_{L_{u0}}(Enc_{L_{v1}}(L_{wg(0,1)})) \\ Enc_{L_{u1}}(Enc_{L_{v0}}(L_{wg(1,0)})) \\ Enc_{L_{u1}}(Enc_{L_{v1}}(L_{wg(1,1)})) \end{array}}$$

This enables one to obtain the label on w according to g given labels on u and v.

- We publish as the garbled circuit (GC) the tables for each gate and the mapping $(L_{out0}, L_{out1}) \to (0, 1)$ for the labels of the output wire.

- $\mathsf{GarbleInp}(x)$: $gx = (L_{1x_1}, \cdots, L_{nx_n})$
- $\mathsf{GarbleEval}(GC, gx)$:
  - For any gate, given a label for each of its input wires, we can find the label that its output wire should have by decrypting the appropriate entry in the gate's table. We will know which of the four to decrypt given the input labels because the encryption scheme we use is checkable. This way, we can obtain a label for the gate's output wire that corresponds to computing the gate on the inputs corresponding to the input labels.
  - To begin with, gx gives us labels on the input wires of the circuit corresponding to the input values. This lets us 'evaluate' gates whose inputs are among these input wires and obtain the labels for their output wires. Proceeding thus in a topological fashion, we can find the label for the output wire of the circuit. At this point, as we know which of L_{out0} and L_{out1} is which, we may learn the output of the circuit.

The correctness of the above garbling scheme is obvious given correctness of the encryption scheme used. It is also efficient because computing gx consists simply of picking L_{ix_i}'s.

That this is secure is proven in the appendix, based on [LP09].

2 Identity-based Encryption

In a traditional public-key encryption scheme, in order to encrypt information such that only a specific user is able to decrypt it, one needs knowledge of a public encryption key that the user has generated some time in the past and has a decryption key for. This is generally realised by the establishment of a public-key infrastructure that allows a lookup of such a key given the identity of the user. But what if it were possible to encrypt data for a specific user given only the identity of the user? This is what Identity-Based Encryption (IBE) lets one do.

The standard model for IBE is one where there is a trusted authority that, given an identity (and an authentication that it is of the party submitting the query), produces a key that may be used to decrypt any ciphertext that was encrypted using that identity. To encrypt to a user, one now needs to know only the user's identity, which could be anything - a phone number or an email address, which one needs to know to communicate with the user anyway.

Definition 2. *An Identity-Based Encryption Scheme consists of the following algorithms:*

- $\mathsf{Setup}(1^\lambda)$ *takes a security parameter* 1^λ *and outputs* MPK*, the Master Public Key, which is released, and* MSK*, the Master Secret Key, which is used to generate other secret keys.*
- $\mathsf{KeyGen}(\mathsf{MSK}, \mathsf{id})$ *generates the secret key* sk_{id} *for identity* id *using the master secret key* MSK*.*
- $\mathsf{Enc}(\mathsf{MPK}, \mathsf{id}, \mathsf{M})$ *outputs an encryption* CT_{id} *of* M *for the identity* id *using the master public key* MPK*.*
- $\mathsf{Dec}(\mathsf{sk_{id}}, \mathsf{CT_{id'}})$ *outputs a decryption of* $CT_{id'}$ *using secret key* sk_{id} *if* $id = id'$*, and* $\perp$ *otherwise.*

Identity-based encryption is significant in our context because it is functional encryption for the following class of functions:

$$f_{id}(id', M) = \begin{cases} (id, M) & , if\ id = id' \\ (id', \perp) & , otherwise \end{cases}$$

We now see how to construct an IBE scheme starting from Regev's PKE scheme based on LWE. This, as seen in earlier lectures, is as follows:

- KeyGen:
  - $sk = s \xleftarrow{R} \mathbb{Z}_q^n$
  - $pk = (A, b = As + e)$, where $A \xleftarrow{R} \mathbb{Z}_q^{m \times n}$, $e \xleftarrow{R} \chi^m$, with $m >> n$.
- $\mathsf{Enc}(pk = (A, b), \mu)$:
  - $CT = (rA, rb + m\lceil q/2 \rfloor)$, where $r \xleftarrow{R} \{0,1\}^m$.
- $\mathsf{Dec}(sk = s, CT = (c_1, c_2))$:
  - $c_2 - c_1 s = (rb + m\lceil q/2 \rfloor) - (rA)s = re + m\lceil q/2 \rfloor \approx m\lceil q/2 \rfloor$
  - If $|re| < q/4$, m may be obtained from $(c_2 - c_1 s)$ by rounding it to 0 or $\lceil q/2 \rfloor$, whichever is closer, giving $m = 0$ and $m = 1$, respectively.

A natural approach to constructing an IBE scheme from here would be to have the matrix A as the master public key. A public hash function could be used to hash a given identity to a vector in $\mathbb{Z}_q^m$ to serve as b for that identity in the above description. But now we face a problem that was perhaps the primary obstruction to realising IBE - how does one generate a secret key corresponding to a given public key?

In our case, given A and b, we need to find s such that $As \approx b$. It is obvious that knowledge of A and b alone will not suffice, as then the above encryption scheme would have been insecure. But first, does such an s even exist?

To understand this better, consider the lattice defined by the columns of A. $As + e$ is a point close to the point As in this lattice. In order for an s to exist such that $b = As + e$ for some small e, b has to be close to a lattice point. As in our case b is chosen to be the hash of an identity and could be anything, it is very likely that it is not close to a lattice point and hence cannot be used as a public key. In order to remedy this, we shift to the 'dual' of this scheme (as observed in [GPV08]), which is as follows.

- KeyGen:
  - $sk = r \xleftarrow{R} \{0,1\}^m$
  - $pk = (A, y = rA)$, where $A \xleftarrow{R} \mathbb{Z}_q^{m \times n}$, with $m >> n \log q$.
- $\mathsf{Enc}(pk = (A, y), \mu)$:
  - $CT = (As + e, ys + e' + \mu\lceil q/2 \rfloor)$, where $s \xleftarrow{R} \mathbb{Z}_q^n$, $e, e' \xleftarrow{R} \chi^m$.
- $Dec(sk = r, CT = (c_1, c_2))$:
  - $c_2 - rc_1 = (ys + e' + \mu\lceil q/2 \rfloor) - r(As + e) = (e' - re) + \mu\lceil q/2 \rfloor \approx \mu\lceil q/2 \rfloor$.

Here $y = rA$ can be almost anything. This is because r has m bits of entropy and rA has at most $n \log q$ bits. As $m >> n \log q$, the Leftover Hash Lemma tells us that rA is statistically close to random. So given y and A, it is quite likely that r exists such that $y = rA$.

Given that such an r exists, how do we find it? Note that we need r to be small (even if not in $\{0,1\}^m$) for the decryption to work. If this were not the case, r could have be found easily by Gaussian elimination. But what now?

What we necessarily need to do is to invert $f_A(r) = rA$ to a small r in the domain. As such, this is believed to be hard. It is left as an exercise to show that if this can be done for an f_A, then it is easy to break LWE w.r.t. A.

Hence, if we obtain y from an identity in our IBE scheme, we shall need some help to be able to find the corresponding secret key. What exactly is this deus ex machina? That is a matter for another week.

References

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *Proceedings of the 2012 ACM conference on Computer and communications security*, CCS '12, pages 784–796, New York, NY, USA, 2012. ACM.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, STOC '08, pages 197–206, New York, NY, USA, 2008. ACM.

[LP09] Yehuda Lindell and Benny Pinkas. A proof of security of yao's protocol for two-party computation. *J. Cryptology*, 22(2):161–188, 2009.

[Yao82] Andrew C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 160–164, Washington, DC, USA, 1982. IEEE Computer Society.

A Security of Yao's Garbling Scheme

(We show here how to construct a simulator for Yao's Garbling Scheme whose output is indistinguishable from (GC, gx), as required for the security of the scheme. This proof is adapted from [LP09], as are several of the following definitions.)

To begin with, we shall need an additional notion of security (along with checkability as described earlier) for a secret-key encryption scheme that we shall be using. We use the following notation for a while.

- $\mathsf{Enc}'(k_0, k_1, m)$ denotes the double encryption $\mathsf{Enc}_{k_0}(\mathsf{Enc}_{k_1}(m))$.
- $Enc'(k, ., .)$ is the oracle that takes key k', message m' and outputs $\mathsf{Enc}'(k, k', m)$. $Enc'(., k, .)$ is defined similarly.

We define the following experiment.

$\mathbf{Expt}^{\mathbf{double}}_{\mathbf{A}}(\mathbf{n}, \sigma)$:

1. Adversary A is invoked on input 1^n and outputs two keys k_0 and k_1 of length n and two triples of messages (x_0, y_0, z_0) and (x_1, y_1, z_1), where all messages are of the same length.
2. Two keys $k'_0, k'_1 \leftarrow \mathsf{KeyGen}(1^n)$ are chosen by the key generation algorithm of the encryption scheme.
3. A is given the challenge $(\mathsf{Enc}'(k_0, k'_1, x_\sigma), \mathsf{Enc}'(k'_0, k_1, y_\sigma), \mathsf{Enc}'(k'_0, k'_1, z_\sigma))$ and access to encryption oracles $Enc(k'_0, ., .)$ and $Enc(., k'_1, .)$.
4. A outputs guess b for σ.

Definition 3. *An encryption scheme is secure under* chosen double encryption *if, for every non-uniform probabilistic polynomial-time machine A, every polynomial p, and all sufficiently large n,*

$$\left| Pr\left[\text{Expt}_A^{double}(n,1)=1\right] - Pr\left[\text{Expt}_A^{double}(n,0)=1\right]\right| < \frac{1}{p(n)}$$

Lemma 4 ([LP09])**.** *Any secret-key encryption scheme that has indistinguishable encryptions under chosen plaintext attack in the presence of nonuniform adversaries is secure under chosen double encryption.*

A proof of the above lemma (which the margin here is too small to contain) may be found in [LP09].

We now construct a simulator SIM that given $|C|$, $|x|$ and $C(x)$, outputs (GC', gx') that is indistinguishable from a legitimate (GC, gx) pair produced by GarbleCkt and GarbleInp. Note that by our assumption, $|C|$ completely determines the topology of the circuit.

- For each wire w, SIM picks at random labels L_{w0}, L'_{w1}.
- For each gate $(w; u, v)$, in place of the table in the actual GC, SIM publishes encryptions of L_{w0} under all pairs of labels of u and v, in random order.

$Enc_{L_{u0}}(Enc_{L_{v0}}(L_{w0}))$
$Enc_{L_{u0}}(Enc_{L_{v1}}(L_{w0}))$
$Enc_{L_{u1}}(Enc_{L_{v0}}(L_{w0}))$
$Enc_{L_{u1}}(Enc_{L_{v1}}(L_{w0}))$

- For the output mapping, SIM maps L_{out0} to $C(x)$ (which it knows) and L_{out1} to $(1 - C(x))$.
- In place of gx, SIM publishes $gx' = (L_{10}, \cdots, L_{n0})$.

To begin with, it is easy to see that running GarbleEval on (GC', gx') produces the correct output, as we have mapped L_{out0} (which is the only possible decryption of any entry in the output gate's table) to $C(x)$. In order to show that the distribution of (GC', gx') is indistinguishable from that of (GC, gx), we define the following hybrids on the garbling of circuits, assuming a topological ordering of gates.

Hybrids: In hybrid H_i, the tables for all gates upto gate i (including i) are published according to SIM, and the tables for the remaining gates are published according to GarbleCkt.

That is, in the tables of all gates up to i, all four encryptions are of the same label, and for the rest the encryptions obey the functionality of the gate. It is straightforward to see that $H_0 = GC'$ and $H_{|C|} = GC$. Also note that gx and gx' are already distributed identically (labels of the input wires).

By standard hybrid arguments, we have that for any PPT adversary A that distinguished between GC and GC' with non-negligible probability, there exists i and polynomial p such that:

$$|Pr\left[A(H_i)=1\right] - Pr\left[A(H_{i-1})=1\right]| > \frac{1}{p(n)}$$

Without loss of generality, let A be such that $Pr\left[A(H_{i-1})=1\right] < Pr\left[A(H_i)=1\right]$. We use this ability of A to construct a challenger that breaks the *chosen double encryption* security of the encryption scheme used. The challenger A' deals with $\text{Expt}_{A'}^{double}(n,\sigma)$ as follows:

1. A receives 1^n.
2. Consider a randomly picked circuit C of size $poly(1^n)$ in the family of circuits that A breaks the security of the garbling scheme in as above, and let i be such that A dintinguishes between H_i and H_{i+1} for this circuit size.

3. Let the input wires of gate i of C be u and v, and the output wire be w, and the chosen labels of w be L_{w0} and L_{w1}. A' chooses these and also labels for all wires except u and v.

4. A' outputs as keys k_0 and k_1, chosen at random, and sets $L_{u0} = k_0$ and $L_{v0} = k_1$.

5. The keys chosen by the encryption algorithm challenging A' are taken to be $L_{u1} = k'_0$ and $L_{v1} = k'_1$, respectively. Note that A' is not given these keys.

6. As the triples A' outputs $(L_{wg_i(0,1)}, L_{wg_i(1,0)}, L_{wg_i(1,1)})$ and $(L_{wg_i(0,0)}, L_{wg_i(0,0)}, L_{wg_i(0,0)})$.

7. A' receives challenge $(\mathsf{Enc}'(L_{u0}, L_{v1}, x), \mathsf{Enc}'(L_{u1}, L_{v0}, y), \mathsf{Enc}'(L_{u1}, L_{v1}, z))$, and access to encryption oracles $Enc(L_{u1}, ., .)$ and $Enc(., L_{v1}, .)$. Denote the challenge by (c_1, c_2, c_3).

8. A' now constructs a hybrid circuit as follows:

   - Any gate before i that does not involve the wires u or v is constructed according to SIM (with the table consisting of four encryptions of the same output label).
   - Any gate after i that does not involve wires u or v (but may have w as input) is constructed according to $\mathsf{GarbleCkt}$ (with the table reflecting the functionality of the gate).
   - The gates before i with output u or v are constructed according to SIM. This may be done because A' knows L_{u0} and L_{v0}, and encryptions in SIM are of these only.
   - For any gate that has u or v (but not both) as input, the encryptions in the table (as per SIM if this is before i, as per $\mathsf{GarbleCkt}$ otherwise) may be constructed using the oracles $Enc(L_{u1}, ., .)$ and $Enc(., L_{v1}, .)$. This is because the only labels A' does not know are L_{u1} and L_{v1}, and hence the other key involved in the double encryption is known. Also, as all our gates are symmetric, the order of the wires does not matter, and $Enc(., L_{v1}, .)$ may be used in the place of $Enc(L_{v1}, ., .)$.
   - The table for gate i (with inputs u and v) is (c, c_1, c_2, c_3), where $c = \mathsf{Enc}'(L_{u0}, L_{v0}, L_{wg(0,0)})$, which can be computed as L_{u0} and L_{v0} are known to A', and c_1, c_2 and c_3 are from the challenge.

9. A' runs A with this hybrid as the putative garbled circuit and the garbled input as $(L_{10}, \cdots, L_{n0})$, and outputs whatever A outputs as its guess of σ.

The above construction by A' is exactly H_i if $\sigma = 1$, and H_{i-1} if $\sigma = 0$. This is because if $\sigma = 0$, the table of gate g_i consists of encryptions of the labels corresponding to the functionality of the gate, and if $\sigma = 1$ all the encryptions are of L_{w0}. Hence, the advantage A' has in distinguishing between these cases is the same advantage A has in distingishing between H_i and H_{i-1}. Since this is non-negligible, A' breaks security of the encryption scheme under chosen double encryption.

Hence, if the encryption scheme used is secure under chosen double encryption, no adversary can distinguish between H_i and H_{i-1} with non-negligible probability for any i. As gx and gx' are from the same distribution, this means that no adversary can distinguish between (GC, gx) and (GC', gx'), which is output by SIM, with non-negligible probability. This shows that the garbling scheme is secure.