CS 294: Special Topic in Cryptography: Secure Computation — Feb 23, 2016

Lecture 11: Efficiency Optimizations on Garbled Circuits

Instructor: Payman Mohassel (guest lecturer) — *Scribe: Peihan Miao*

1 Yao's Garbled Circuit

Definition 1 *A garbled circuit scheme consists of* $(\mathsf{Garble}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *where*

$$\begin{aligned}
&\mathsf{Garble}\left(1^\kappa, \mathsf{C}\right) \to \left(\tilde{\mathsf{C}}, E, D\right)\\
&\mathsf{Enc}\left(E, x\right) \to \tilde{x}\\
&\mathsf{Eval}(\tilde{\mathsf{C}}, \tilde{x}) \to \widetilde{\mathsf{C}(x)}\\
&\mathsf{Dec}\left(\widetilde{\mathsf{C}(x)}, D\right) \to \mathsf{C}(x)
\end{aligned}$$

and the following properties hold:

- ***Correctness:***

  $$\mathbb{P}\left[\mathsf{Garble}\left(1^\kappa, \mathsf{C}\right) \to \left(\tilde{\mathsf{C}}, E, D\right), \mathsf{Enc}\left(E, x\right) \to \tilde{x}, \mathsf{Eval}(\tilde{\mathsf{C}}, \tilde{x}) \to \widetilde{\mathsf{C}(x)} : \mathsf{C}(x) = \mathsf{Dec}\left(\widetilde{\mathsf{C}(x)}, D\right)\right] = 1.$$

- ***Privacy:*** *There exists a PPT simulator* $\mathcal{S}$ *such that for any* C, x,

  $$\mathcal{S}\left(1^\kappa, \mathsf{C}, \mathsf{C}(x)\right) \approx_c \left(\tilde{\mathsf{C}}, \tilde{x}, D\right)$$

  where $\mathsf{Garble}\left(1^\kappa, \mathsf{C}\right) \to \left(\tilde{\mathsf{C}}, E, D\right), \mathsf{Enc}\left(E, x\right) \to \tilde{x}.$

- ***Output Authenticity:*** *An adversary who learns* $\tilde{\mathsf{C}}$ *and* $\tilde{x}$ *should be unable to produce a valid garbled output different from* $\widetilde{\mathsf{C}(x)}$. *Note that this is an optional property which doesn't necessarily hold.*

Recall the computation and communication complexity of Yao's garbled circuit. Consider an AND gate with input wire a, b and output wire c. Each wire has two labels $\left\{\mathsf{k}_a^0, \mathsf{k}_a^1\right\}, \left\{\mathsf{k}_b^0, \mathsf{k}_b^1\right\}, \left\{\mathsf{k}_c^0, \mathsf{k}_c^1\right\}$. The garbled gate consists of the following four ciphertexts in a randomly permuted order:

$$\begin{aligned}
&\mathsf{Enc}_{\mathsf{k}_a^0}\left(\mathsf{Enc}_{\mathsf{k}_b^0}\left(\mathsf{k}_c^0 || 0^\kappa\right)\right)\\
&\mathsf{Enc}_{\mathsf{k}_a^0}\left(\mathsf{Enc}_{\mathsf{k}_b^1}\left(\mathsf{k}_c^0 || 0^\kappa\right)\right)\\
&\mathsf{Enc}_{\mathsf{k}_a^1}\left(\mathsf{Enc}_{\mathsf{k}_b^0}\left(\mathsf{k}_c^0 || 0^\kappa\right)\right)\\
&\mathsf{Enc}_{\mathsf{k}_a^1}\left(\mathsf{Enc}_{\mathsf{k}_b^1}\left(\mathsf{k}_c^1 || 0^\kappa\right)\right)
\end{aligned}$$

Note that the "$||0^\kappa$" part achieves verifiable decryption. When evaluating the garbled circuit, one needs to try decrypting all four ciphertexts and see which one gives a valid label.

2 Optimizations

We introduce several techniques to reduce both the computation and communication complexity of Yao's garbling scheme.

2.1 Point-And-Permute [BMR90, MNP+04]

Take the AND gate as an example. Along with each wire the garbler assign a random bit and attach to the labels, in particular, $\left\{\mathsf{k}_a^0||r_a, \mathsf{k}_a^1||\overline{r_a}\right\}, \left\{\mathsf{k}_b^0||r_b, \mathsf{k}_b^1||\overline{r_b}\right\}, \left\{\mathsf{k}_c^0||r_c, \mathsf{k}_c^1||\overline{r_c}\right\}$. The random bits play the role of random permutating the ciphertexts. More precisely, now the garbled gate consists of the following four ciphertexts:

$$\text{when evaluator gets } \mathsf{k}_a^{r_a}||0, \mathsf{k}_b^{r_b}||0 : \mathsf{Enc}_{\mathsf{k}_a^{r_a}}\left(\mathsf{Enc}_{\mathsf{k}_b^{r_b}}\left(\mathsf{k}_c^{r_a \wedge r_b}||\left(r_a \wedge r_b\right) \oplus r_c\right)\right)$$
$$\text{when evaluator gets } \mathsf{k}_a^{r_a}||0, \mathsf{k}_b^{\overline{r_b}}||1 : \mathsf{Enc}_{\mathsf{k}_a^{r_a}}\left(\mathsf{Enc}_{\mathsf{k}_b^{\overline{r_b}}}\left(\mathsf{k}_c^{r_a \wedge \overline{r_b}}||\left(r_a \wedge \overline{r_b}\right) \oplus r_c\right)\right)$$
$$\text{when evaluator gets } \mathsf{k}_a^{\overline{r_a}}||1, \mathsf{k}_b^{r_b}||0 : \mathsf{Enc}_{\mathsf{k}_a^{\overline{r_a}}}\left(\mathsf{Enc}_{\mathsf{k}_b^{r_b}}\left(\mathsf{k}_c^{\overline{r_a} \wedge r_b}||\left(\overline{r_a} \wedge r_b\right) \oplus r_c\right)\right)$$
$$\text{when evaluator gets } \mathsf{k}_a^{\overline{r_a}}||1, \mathsf{k}_b^{\overline{r_b}}||1 : \mathsf{Enc}_{\mathsf{k}_a^{\overline{r_a}}}\left(\mathsf{Enc}_{\mathsf{k}_b^{\overline{r_b}}}\left(\mathsf{k}_c^{\overline{r_a} \wedge \overline{r_b}}||\left(\overline{r_a} \wedge \overline{r_b}\right) \oplus r_c\right)\right)$$

The random bits can point to the evaluator which ciphertext he should decrypt without revealing whether it's a zero-label or one-label. Therefore the computation complexity of the evaluator is decreased by a factor of 4 per gate.

2.2 Free-XOR [KS08]

In the original garbling scheme and point-and-permute optimization, the communication complexity is the same for an AND gate and an XOR gate, and so do computation complexity. In this section we introduce a technique which can get us no communication cost for XOR gates. The garbler first samples a random "global secret" Δ, and makes the one-label for each wire be the XOR of its corresponding zero-label and Δ, namely $\mathsf{k}_a^1 = \mathsf{k}_a^0 \oplus \Delta, \mathsf{k}_b^1 = \mathsf{k}_b^0 \oplus \Delta, \mathsf{k}_c^1 = \mathsf{k}_c^0 \oplus \Delta$. If the garbler further makes $\mathsf{k}_c^0 = \mathsf{k}_a^0 \oplus \mathsf{k}_b^0$, then apparently $\mathsf{k}_c^{\alpha \oplus \beta} = \mathsf{k}_a^\alpha \oplus \mathsf{k}_b^\beta$, and the communication cost for this XOR gate is 0.

2.3 Garbled Row Reduction (GRR3) [NPS99]

The idea of this optimization is to make $\mathsf{k}_c^0 = \mathcal{H}(\mathsf{k}_a^0||\mathsf{k}_b^0)$ where $\mathcal{H}(\cdot)$ is a hash function, so that we can get rid of one ciphertext per (AND/XOR) gate. One needs the idea of point-and-permute to point out when to apply the hash function. Furthermore, this technique is compatible with Free-XOR [KS08].

2.4 Garbled Row Reduction (GRR2) [PSSW09]

The goal of this section is to get rid of two ciphertext per (AND/XOR) gate instead of one. Take an AND gate as an example. First the garbler compute the following:

$$\begin{aligned} k^1 &\leftarrow \mathcal{H}\left(k_a^0 || k_b^0\right) \\ k^2 &\leftarrow \mathcal{H}\left(k_a^0 || k_b^1\right) \\ k^3 &\leftarrow \mathcal{H}\left(k_a^1 || k_b^0\right) \\ k^4 &\leftarrow \mathcal{H}\left(k_a^1 || k_b^1\right) \end{aligned}$$

Then find a polynomial $p(\cdot)$ of degree 2 such that $p(1) = k^1, p(2) = k^2, p(3) = k^3$. Define $k_c^0 = p(0)$. Find another polynomial $q(\cdot)$ of degree 2 such that $q(4) = k^4, q(5) = p(5), q(6) = p(6)$, and define $k_c^1 = q(0)$. The garbled gate consists of only two elements $p(5)$ and $p(6)$. Note that the point-and-permute technique is still needed. It works similarly for XOR gates.

2.5 FleXOR [KMR14]

When we apply the GRR2 optimization technique, it no longer holds that for every wire one-label is an XOR of the zero-label and a global secret Δ. Therefore it is not compatible with Free-XOR. FleXOR aims to apply Free-XOR as well, the idea being to make the differences between one-label and zero-label for each wire identical. More specifically, consider an XOR gate with input labels $\left\{\mathsf{k}_a^0, \mathsf{k}_a^0 \oplus \Delta_a\right\}, \left\{\mathsf{k}_b^0, \mathsf{k}_b^0 \oplus \Delta_b\right\}$ and output labels $\left\{\mathsf{k}_c^0, \mathsf{k}_c^0 \oplus \Delta_c\right\}$. If $\Delta_a \neq \Delta_c$, then the garbler provides two ciphertexts $\mathsf{Enc}_{\mathsf{k}_a^0}\left(\widetilde{\mathsf{k}_a^0}\right), \mathsf{Enc}_{\mathsf{k}_a^1}\left(\widetilde{\mathsf{k}_a^1}\right)$ such that $\widetilde{\mathsf{k}_a^1} = \widetilde{\mathsf{k}_a^0} \oplus \Delta_c$. In such a way the evaluator can transform $\left\{\mathsf{k}_a^0, \mathsf{k}_a^0 \oplus \Delta_a\right\}$ to a new pair of labels $\left\{\widetilde{\mathsf{k}_a^0}, \widetilde{\mathsf{k}_a^0} \oplus \Delta_c\right\}$. By applying the GRR3 trick again to make $\widetilde{\mathsf{k}_a^0} = \mathcal{H}\left(\mathsf{k}_a^0\right)$ we can get rid of one ciphertext. The same technique should be done for $\left\{\mathsf{k}_b^0, \mathsf{k}_b^0 \oplus \Delta_b\right\}$ as well. Thus for the garbling of each XOR gate it contains 0, 1, or 2 ciphertexts.

2.6 Half-gates [ZRE15]

Stepping back to Free-XOR where for each wire $\mathsf{k}^1 = \mathsf{k}^0 \oplus \Delta$. Let's see what we can do for an AND gate with input wires a, b and output wire c.

- One half gate: if the garbler knows the value of a, he only needs to provide two ciphertexts: $\mathcal{H}\left(\mathsf{k}_b^0\right) \oplus \mathsf{k}_c^0$ and $\mathcal{H}\left(\mathsf{k}_b^0 \oplus \Delta\right) \oplus \mathsf{k}_c^0 \oplus a\Delta$. The first one can be further thrown away by making $\mathsf{k}_c^0 = \mathcal{H}\left(\mathsf{k}_b^0\right)$.
- The other half gate: if the evaluator knows the value of a, he only needs two ciphertexts from the garbler: $\mathcal{H}\left(\mathsf{k}_a^0\right) \oplus \mathsf{k}_c^0$ and $\mathcal{H}\left(\mathsf{k}_a^0 \oplus \Delta\right) \oplus \mathsf{k}_c^0 \oplus \mathsf{k}_b^0$, and the first one can be thrown away by setting $\mathsf{k}_c^0 = \mathcal{H}\left(\mathsf{k}_a^0\right)$.

A gate $c = a \wedge b$ can be written as $c = (a \wedge r) \oplus (a \wedge (b \oplus r))$, where in the first half gate $(a \wedge r)$ the garbler knows r, and in the second half gate $(a \wedge (b \oplus r))$ the evaluator knows $b \oplus r$ from the point-and-permute random bit. Therefore, each AND gate only needs two ciphertexts, and Free-XOR still holds.

References

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 503–513. ACM, 1990.

[KMR14] Vladimir Kolesnikov, Payman Mohassel, and Mike Rosulek. Flexor: Flexible garbling for xor gates that beats free-xor. In *Advances in Cryptology–CRYPTO 2014*, pages 440–457. Springer, 2014.

[KS08] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free xor gates and applications. In *Automata, Languages and Programming*, pages 486–498. Springer, 2008.

[MNP+04] Dahlia Malkhi, Noam Nisan, Benny Pinkas, Yaron Sella, et al. Fairplay — secure two-party computation system. In *USENIX Security Symposium*, volume 4. San Diego, CA, USA, 2004.

[NPS99] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 129–139. ACM, 1999.

[PSSW09] Benny Pinkas, Thomas Schneider, Nigel P Smart, and Stephen C Williams. Secure two-party computation is practical. In *Advances in Cryptology–ASIACRYPT 2009*, pages 250–267. Springer, 2009.

[ZRE15] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole. In *Advances in Cryptology-EUROCRYPT 2015*, pages 220–250. Springer, 2015.