

Reach Restricted Reactive Program Obfuscation

And its Application to MA-ABE

Kaartik **B**hushan, Sai Lakshmi Bhavana **O**bbattu, Manoj **P**rabhakaran,
Rajeev Raghunath

April 14, 2024

Some Surprising Results

Laconic Oblivious Transfer from DDH [CDG⁺17]

Identity Based Encryption (IBE) from DDH [DG17a, DG17b]

2-round MPC from 2-round OT [BL17, GS17]

Some Surprising Results

Laconic Oblivious Transfer from DDH [CDG⁺17]

Identity Based Encryption (IBE) from DDH [DG17a, DG17b]

2-round MPC from 2-round OT [BL17, GS17]

Key Ingredient in all the above: Garbled Circuit Chaining

Some Surprising Results

Laconic Oblivious Transfer from DDH [CDG⁺17]

Identity Based Encryption (IBE) from DDH [DG17a, DG17b]

2-round MPC from 2-round OT [BL17, GS17]

Key Ingredient in all the above: Garbled Circuit Chaining

But no common abstraction or framework!

Some Surprising Results

Laconic Oblivious Transfer from DDH [CDG⁺17]

Identity Based Encryption (IBE) from DDH [DG17a, DG17b]

2-round MPC from 2-round OT [BL17, GS17]

Key Ingredient in all the above: Garbled Circuit Chaining

But no common abstraction or framework!

Motivation Behind This Work: Abstract the powerful ingredient driving these results (and beyond)

Some Surprising Results

Laconic Oblivious Transfer from DDH [CDG⁺17]

Identity Based Encryption (IBE) from DDH [DG17a, DG17b]

2-round MPC from 2-round OT [BL17, GS17]

Key Ingredient in all the above: Garbled Circuit Chaining

But no common abstraction or framework!

Motivation Behind This Work: Abstract the powerful ingredient driving these results (and beyond) *as Obfuscation!*

What is Obfuscation?

```
1 #include <stdio.h>
2
3 int main(int argc, char *argv[]) {
4     int i;
5     for (i = 0; i < argc; i++) {
6         printf("%s\n", argv[i]);
7     }
8     return 0;
9 }
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
```

Program Obfuscation:

- keeping secrets in a program

What is Obfuscation?

```
3 int main(int argc, char** argv) {
4     unsigned long long x = 0;
5     unsigned long long y = 0;
6     int i;
7     int j;
8     int k;
9     int l;
10    int m;
11    int n;
12    int o;
13    int p;
14    int q;
15    int r;
16    int s;
17    int t;
18    int u;
19    int v;
20    int w;
21    int x;
22    int y;
23    int z;
24    int aa;
25    int ab;
26    int ac;
27    int ad;
28    int ae;
29    int af;
30    int ag;
31    int ah;
32    int ai;
33    int aj;
34    int ak;
35    int al;
36    int am;
37    int an;
38    int ao;
39    int ap;
40    int aq;
41    int ar;
42    int as;
43    int at;
44    int au;
45    int av;
46    int aw;
47    int ax;
48    int ay;
49    int az;
50    int ba;
51    int bb;
52    int bc;
53    int bd;
54    int be;
55    int bf;
56    int bg;
57    int bh;
58    int bi;
59    int bj;
60    int bk;
61    int bl;
62    int bm;
63    int bn;
64    int bo;
65    int bp;
66    int bq;
67    int br;
68    int bs;
69    int bt;
70    int bu;
71    int bv;
72    int bw;
73    int bx;
74    int by;
75    int bz;
76    int ca;
77    int cb;
78    int cc;
79    int cd;
80    int ce;
81    int cf;
82    int cg;
83    int ch;
84    int ci;
85    int cj;
86    int ck;
87    int cl;
88    int cm;
89    int cn;
90    int co;
91    int cp;
92    int cq;
93    int cr;
94    int cs;
95    int ct;
96    int cu;
97    int cv;
98    int cw;
99    int cx;
100   int cy;
101   int cz;
102   int da;
103   int db;
104   int dc;
105   int dd;
106   int de;
107   int df;
108   int dg;
109   int dh;
110   int di;
111   int dj;
112   int dk;
113   int dl;
114   int dm;
115   int dn;
116   int do;
117   int dp;
118   int dq;
119   int dr;
120   int ds;
121   int dt;
122   int du;
123   int dv;
124   int dw;
125   int dx;
126   int dy;
127   int dz;
128   int ea;
129   int eb;
130   int ec;
131   int ed;
132   int ee;
133   int ef;
134   int eg;
135   int eh;
136   int ei;
137   int ej;
138   int ek;
139   int el;
140   int em;
141   int en;
142   int eo;
143   int ep;
144   int eq;
145   int er;
146   int es;
147   int et;
148   int eu;
149   int ev;
150   int ew;
151   int ex;
152   int ey;
153   int ez;
154   int fa;
155   int fb;
156   int fc;
157   int fd;
158   int fe;
159   int ff;
160   int fg;
161   int fh;
162   int fi;
163   int fj;
164   int fk;
165   int fl;
166   int fm;
167   int fn;
168   int fo;
169   int fp;
170   int fq;
171   int fr;
172   int fs;
173   int ft;
174   int fu;
175   int fv;
176   int fw;
177   int fx;
178   int fy;
179   int fz;
180   int ga;
181   int gb;
182   int gc;
183   int gd;
184   int ge;
185   int gf;
186   int gg;
187   int gh;
188   int gi;
189   int gj;
190   int gk;
191   int gl;
192   int gm;
193   int gn;
194   int go;
195   int gp;
196   int gq;
197   int gr;
198   int gs;
199   int gt;
200   int gu;
201   int gv;
202   int gw;
203   int gx;
204   int gy;
205   int gz;
206   int ha;
207   int hb;
208   int hc;
209   int hd;
210   int he;
211   int hf;
212   int hg;
213   int hh;
214   int hi;
215   int hj;
216   int hk;
217   int hl;
218   int hm;
219   int hn;
220   int ho;
221   int hp;
222   int hq;
223   int hr;
224   int hs;
225   int ht;
226   int hu;
227   int hv;
228   int hw;
229   int hx;
230   int hy;
231   int hz;
232   int ia;
233   int ib;
234   int ic;
235   int id;
236   int ie;
237   int if;
238   int ig;
239   int ih;
240   int ii;
241   int ij;
242   int ik;
243   int il;
244   int im;
245   int in;
246   int io;
247   int ip;
248   int iq;
249   int ir;
250   int is;
251   int it;
252   int iu;
253   int iv;
254   int iw;
255   int ix;
256   int iy;
257   int iz;
258   int ja;
259   int jb;
260   int jc;
261   int jd;
262   int je;
263   int jf;
264   int jg;
265   int jh;
266   int ji;
267   int jj;
268   int jk;
269   int jl;
270   int jm;
271   int jn;
272   int jo;
273   int jp;
274   int jq;
275   int jr;
276   int js;
277   int jt;
278   int ju;
279   int jv;
280   int jw;
281   int jx;
282   int jy;
283   int jz;
284   int ka;
285   int kb;
286   int kc;
287   int kd;
288   int ke;
289   int kf;
290   int kg;
291   int kh;
292   int ki;
293   int kj;
294   int kk;
295   int kl;
296   int km;
297   int kn;
298   int ko;
299   int kp;
300   int kq;
301   int kr;
302   int ks;
303   int kt;
304   int ku;
305   int kv;
306   int kw;
307   int kx;
308   int ky;
309   int kz;
310   int la;
311   int lb;
312   int lc;
313   int ld;
314   int le;
315   int lf;
316   int lg;
317   int lh;
318   int li;
319   int lj;
320   int lk;
321   int ll;
322   int lm;
323   int ln;
324   int lo;
325   int lp;
326   int lq;
327   int lr;
328   int ls;
329   int lt;
330   int lu;
331   int lv;
332   int lw;
333   int lx;
334   int ly;
335   int lz;
336   int ma;
337   int mb;
338   int mc;
339   int md;
340   int me;
341   int mf;
342   int mg;
343   int mh;
344   int mi;
345   int mj;
346   int mk;
347   int ml;
348   int mm;
349   int mn;
350   int mo;
351   int mp;
352   int mq;
353   int mr;
354   int ms;
355   int mt;
356   int mu;
357   int mv;
358   int mw;
359   int mx;
360   int my;
361   int mz;
362   int na;
363   int nb;
364   int nc;
365   int nd;
366   int ne;
367   int nf;
368   int ng;
369   int nh;
370   int ni;
371   int nj;
372   int nk;
373   int nl;
374   int nm;
375   int no;
376   int np;
377   int nq;
378   int nr;
379   int ns;
380   int nt;
381   int nu;
382   int nv;
383   int nw;
384   int nx;
385   int ny;
386   int nz;
387   int oa;
388   int ob;
389   int oc;
390   int od;
391   int oe;
392   int of;
393   int og;
394   int oh;
395   int oi;
396   int oj;
397   int ok;
398   int ol;
399   int om;
400   int on;
401   int oo;
402   int op;
403   int oq;
404   int or;
405   int os;
406   int ot;
407   int ou;
408   int ov;
409   int ow;
410   int ox;
411   int oy;
412   int oz;
413   int pa;
414   int pb;
415   int pc;
416   int pd;
417   int pe;
418   int pf;
419   int pg;
420   int ph;
421   int pi;
422   int pj;
423   int pk;
424   int pl;
425   int pm;
426   int pn;
427   int po;
428   int pp;
429   int pq;
430   int pr;
431   int ps;
432   int pt;
433   int pu;
434   int pv;
435   int pw;
436   int px;
437   int py;
438   int pz;
439   int qa;
440   int qb;
441   int qc;
442   int qd;
443   int qe;
444   int qf;
445   int qg;
446   int qh;
447   int qi;
448   int qj;
449   int qk;
450   int ql;
451   int qm;
452   int qn;
453   int qo;
454   int qp;
455   int qq;
456   int qr;
457   int qs;
458   int qt;
459   int qu;
460   int qv;
461   int qw;
462   int qx;
463   int qy;
464   int qz;
465   int ra;
466   int rb;
467   int rc;
468   int rd;
469   int re;
470   int rf;
471   int rg;
472   int rh;
473   int ri;
474   int rj;
475   int rk;
476   int rl;
477   int rm;
478   int rn;
479   int ro;
480   int rp;
481   int rq;
482   int rr;
483   int rs;
484   int rt;
485   int ru;
486   int rv;
487   int rw;
488   int rx;
489   int ry;
490   int rz;
491   int sa;
492   int sb;
493   int sc;
494   int sd;
495   int se;
496   int sf;
497   int sg;
498   int sh;
499   int si;
500   int sj;
501   int sk;
502   int sl;
503   int sm;
504   int sn;
505   int so;
506   int sp;
507   int sq;
508   int sr;
509   int ss;
510   int st;
511   int su;
512   int sv;
513   int sw;
514   int sx;
515   int sy;
516   int sz;
517   int ta;
518   int tb;
519   int tc;
520   int td;
521   int te;
522   int tf;
523   int tg;
524   int th;
525   int ti;
526   int tj;
527   int tk;
528   int tl;
529   int tm;
530   int tn;
531   int to;
532   int tp;
533   int tq;
534   int tr;
535   int ts;
536   int tt;
537   int tu;
538   int tv;
539   int tw;
540   int tx;
541   int ty;
542   int tz;
543   int ua;
544   int ub;
545   int uc;
546   int ud;
547   int ue;
548   int uf;
549   int ug;
550   int uh;
551   int ui;
552   int uj;
553   int uk;
554   int ul;
555   int um;
556   int un;
557   int uo;
558   int up;
559   int uq;
560   int ur;
561   int us;
562   int ut;
563   int uu;
564   int uv;
565   int uw;
566   int ux;
567   int uy;
568   int uz;
569   int va;
570   int vb;
571   int vc;
572   int vd;
573   int ve;
574   int vf;
575   int vg;
576   int vh;
577   int vi;
578   int vj;
579   int vk;
580   int vl;
581   int vm;
582   int vn;
583   int vo;
584   int vp;
585   int vq;
586   int vr;
587   int vs;
588   int vt;
589   int vu;
590   int vv;
591   int vw;
592   int vx;
593   int vy;
594   int vz;
595   int wa;
596   int wb;
597   int wc;
598   int wd;
599   int we;
600   int wf;
601   int wg;
602   int wh;
603   int wi;
604   int wj;
605   int wk;
606   int wl;
607   int wm;
608   int wn;
609   int wo;
610   int wp;
611   int wq;
612   int wr;
613   int ws;
614   int wt;
615   int wu;
616   int wv;
617   int ww;
618   int wx;
619   int wy;
620   int wz;
621   int xa;
622   int xb;
623   int xc;
624   int xd;
625   int xe;
626   int xf;
627   int xg;
628   int xh;
629   int xi;
630   int xj;
631   int xk;
632   int xl;
633   int xm;
634   int xn;
635   int xo;
636   int xp;
637   int xq;
638   int xr;
639   int xs;
640   int xt;
641   int xu;
642   int xv;
643   int xw;
644   int xx;
645   int xy;
646   int xz;
647   int ya;
648   int yb;
649   int yc;
650   int yd;
651   int ye;
652   int yf;
653   int yg;
654   int yh;
655   int yi;
656   int yj;
657   int yk;
658   int yl;
659   int ym;
660   int yn;
661   int yo;
662   int yp;
663   int yq;
664   int yr;
665   int ys;
666   int yt;
667   int yu;
668   int yv;
669   int yw;
670   int yx;
671   int yy;
672   int yz;
673   int za;
674   int zb;
675   int zc;
676   int zd;
677   int ze;
678   int zf;
679   int zg;
680   int zh;
681   int zi;
682   int zj;
683   int zk;
684   int zl;
685   int zm;
686   int zn;
687   int zo;
688   int zp;
689   int zq;
690   int zr;
691   int zs;
692   int zt;
693   int zu;
694   int zv;
695   int zw;
696   int zx;
697   int zy;
698   int zz;
699 }
```

Program Obfuscation:

- keeping secrets in a program
- even against an adversary that captures the entire computer on which it is run
- without any trusted hardware

Subtle to formalize

Different notions of Obfuscation

Virtual Black-Box Obfuscation [BGI⁺01]

Indistinguishability Obfuscation (iO) [BGI⁺01, JLS20]

Average Case Obfuscation [HRsV07]

Virtual Grey Box Obfuscation [BCTKP14]

Differing Inputs Obfuscation [ABG⁺13] and public-coin DiO [IPS14]

Require strong assumptions in general (if not impossible)

Different notions of Obfuscation

Virtual Black-Box Obfuscation [BGI⁺01]

Indistinguishability Obfuscation (iO) [BGI⁺01, JLS20]

Average Case Obfuscation [HRsV07]

Virtual Grey Box Obfuscation [BCTKP14]

Differing Inputs Obfuscation [ABG⁺13] and public-coin DiO [IPS14]

Require strong assumptions in general (if not impossible)

- Obfuscation achievable from standard assumptions, when programs are sampled in a customized fashion:
 - Obfuscation for Re-Encryption [HRsV07] - from DDH
 - Obfuscation for Evasive Functions [BBC⁺13] - from DDH variant
 - Obfuscation for Compute-and-Compare functions [WZ17] - from LWE
 - **In this work:** For programs sampled interactively, enforcing a restriction on what information the adversary has about its contents.

A Motivating Example

A program that contains a message and an encryption public-key PK .
If a valid decryption key SK is given as input, it outputs the message.

A Motivating Example

A program that contains a message and an encryption public-key PK .
If a valid decryption key SK is given as input, it outputs the message.

Naïve idea: *Encrypting the message using PK will be an obfuscation of this program!*

A Motivating Example

A program that contains a message and an encryption public-key PK . If a valid decryption key SK is given as input, it outputs the message.

Naïve idea: *Encrypting the message using PK will be an obfuscation of this program!*

Can be turned into a valid notion of obfuscation:

Interactive sampling of the program: (PK, SK) pairs are generated secretly. Each SK can be published fully, or not revealed at all, as requested by the adversary.

Simulation of Obfuscated Program: If SK published, an adversary is allowed to learn the message — from which a valid ciphertext can be constructed. If SK not published, a random ciphertext is a valid simulation of the real ciphertext.

Conversely, such an obfuscation yields PKE.

A definition that formalizes similar seemingly naïve ideas of obfuscation

Example: IBE as Obfuscation

Ciphertext is the obfuscation of the following program:

Hardwired: message m , identity id , a signature verification key VK

On input σ : if σ is a valid signature on id w.r.t. VK , output m .

Issue a decryption key for id by simply signing id

Reactive Program

Let Σ , M be the space of states and messages respectively.

Reactive Program $P = (\pi_\alpha, \mu_\beta)$:

Transition function $\pi_\alpha : \Sigma \times \mathcal{X} \rightarrow \Sigma$

Message function $\mu_\beta : \Sigma \rightarrow M$

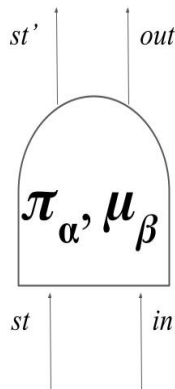
(for some hardwired secrets α, β)

Evaluating a Reactive Program:

$P(st, in)$:

$$st' = \pi_\alpha(st, in)$$

$$out = \mu_\beta(st')$$



Reach-Restricted Reactive Program

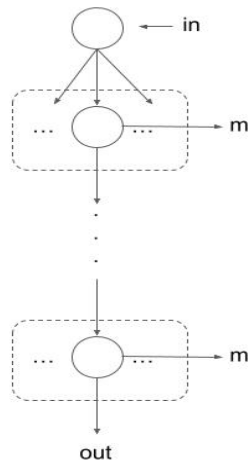
We require a partition of the state space,
 $\Sigma = \Sigma_1 \cup \dots \cup \Sigma_n$

A typical example will have:

- $O(\kappa)$ parts
- $2^{O(\kappa)}$ states in each part.

where κ is the security parameter.

The parts should form a tree.



Reach-Restricted Reactive Program

We require a partition of the state space,
 $\Sigma = \Sigma_1 \cup \dots \cup \Sigma_n$

A typical example will have:

- $O(\kappa)$ parts
- $2^{O(\kappa)}$ states in each part.

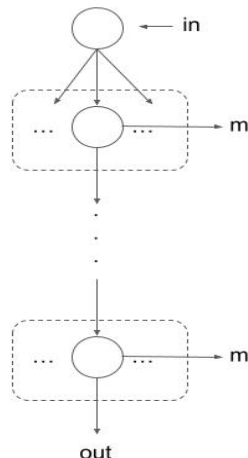
where κ is the security parameter.

The parts should form a tree.

Reach-Restriction:

Adversary can find inputs that take the program to at most one state in each part of the partition

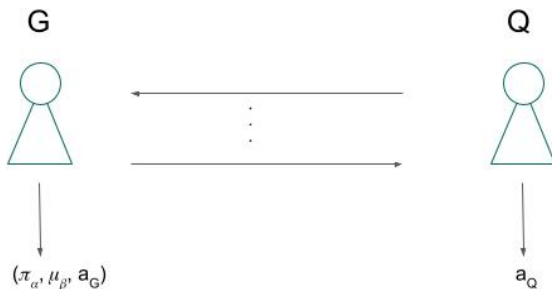
When the program is “sampled properly”



Interactive Sampling

Rules for sampling a program formalized as a class of **Reactive Program Generators**

- A generator G interacts with an adversary Q
- Outputs a reactive program (π_α, μ_β) .
Also auxiliary information a_G, a_Q produced



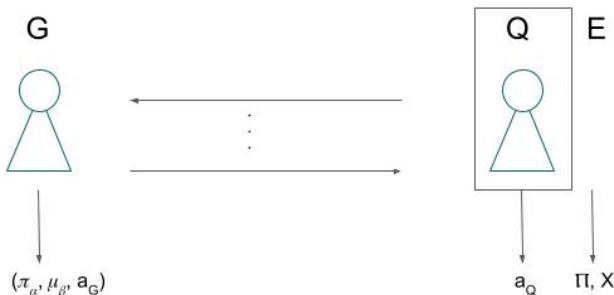
Formalizing Reach Restriction

To which all states can an adversary Q take a reactive program generated by a generator G (even given $(\pi_\alpha, \mu_\beta, a_G; a_Q)$)?

Formalizing Reach Restriction

To which all states can an adversary Q take a reactive program generated by a generator G (even given $(\pi_\alpha, \mu_\beta, a_G; a_Q)$)?

- An extractor E can output all such states.
 - Encoded as an (idealized) reactive program Π and input sequences X for it, s.t. reachable states in π_α are reached in Π using X .
- Can have at most one state in each part in the state-space partition.

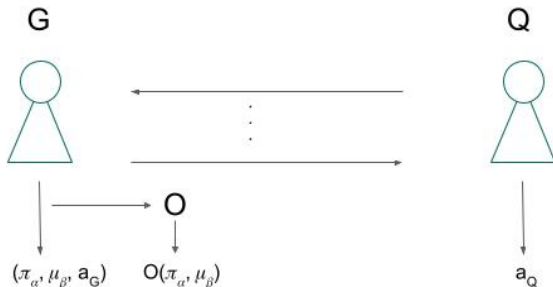


Defining R3PO Security

A Strong Simulation-Based Definition

The Real World:

- G interacts with Q
- output of interaction: $(\pi_\alpha, \mu_\beta, a_G; a_Q)$
- Obfuscator \mathcal{O} outputs $\mathcal{O}(\pi_\alpha, \mu_\beta)$

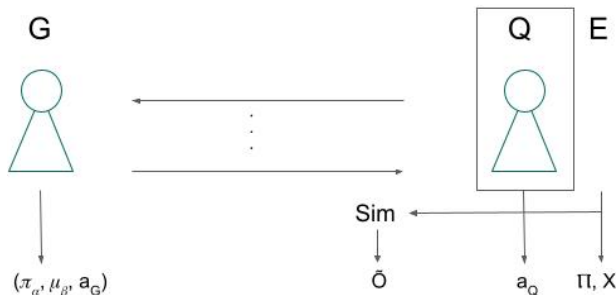


Defining R3PO Security

A Strong Simulation-Based Definition

The Ideal World:

- G interacts with Q
- output of interaction: $(\pi_\alpha, \mu_\beta, a_G; a_Q)$, E outputs Π, X
- $\text{Sim}(\Pi, X, \{\mu_\beta(\text{st}) \mid \text{st} \in \Pi(X)\})$ outputs \tilde{O}



Defining R3PO Security

A Strong Simulation-Based Definition

\mathcal{O} is an R3PO scheme for \mathcal{G} w.r.t. a class of adversaries \mathcal{Q} if, $\forall G \in \mathcal{G}$ and $Q \in \mathcal{Q}$, there exists a simulator Sim s.t. Real World is indistinguishable from Ideal World

$$\left\{ \mathcal{O}(\pi_\alpha, \mu_\beta), a_G, a_Q \right\} \\ \approx \left\{ \text{Sim}\left(\Pi, X, \{\mu_\beta(\text{st}) \mid \text{st} \in \Pi(X)\}\right), a_G, a_Q \right\}$$

Example 1: Commitment Opening R3PO

Commitment Scheme

- $\text{gen}(1^\kappa) \rightarrow \text{crs}$
- $\text{commit}(\text{crs}, m) \rightarrow (c, d)$
- $\text{open}(\text{crs}, c, d) \rightarrow m$

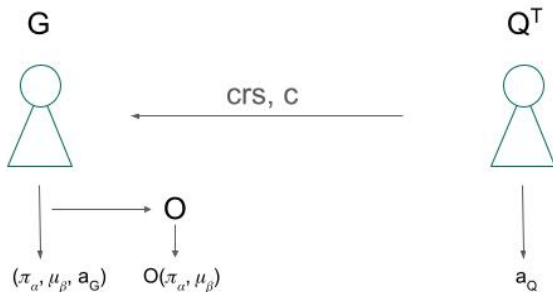
Properties Required:

- 1 Computational Hiding: commitment c does not reveal message m .
- 2 Computational Binding: commitment c can be opened to at most a single message m . Further, there exists an extractor \mathcal{E} that can extract this m .

Example 1: Commitment Opening R3PO

Interaction:

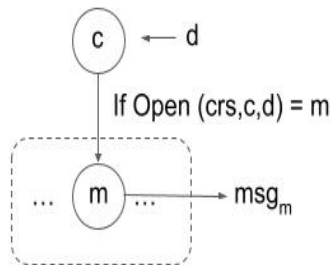
- Q^T gets crs from T and sends crs, c to G .



Example 1: Commitment Opening R3PO

Reactive Program:

$$\pi(\text{st}_c^1, d) = \begin{cases} \text{st}_m^2, & \text{if } \text{open}(\text{crs}, c, d) = m \\ \perp, & \text{else} \end{cases}$$

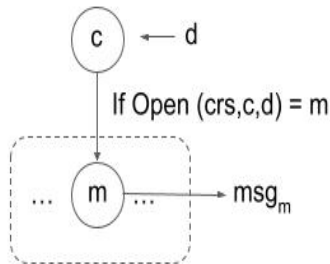


★ Interested in adversaries of the form Q^T that sample crs honestly.

Example 1: Commitment Opening R3PO

Reactive Program:

$$\pi(\text{st}_c^1, d) = \begin{cases} \text{st}_m^2, & \text{if } \text{open}(\text{crs}, c, d) = m \\ \perp, & \text{else} \end{cases}$$



★ Interested in adversaries of the form Q^T that sample crs honestly.

Theorem 1 (Informally)

If the DDH assumption holds, there exists a Commitment scheme and a R3PO for Commitment-Opening.

Signature Scheme

- $\text{gen}(1^\kappa) \rightarrow (\text{vk}, \text{sk})$
- $\text{sign}(\text{sk}, m) \rightarrow \tau$
- $\text{verify}(\text{vk}, m, \tau) \rightarrow \{0, 1\}$

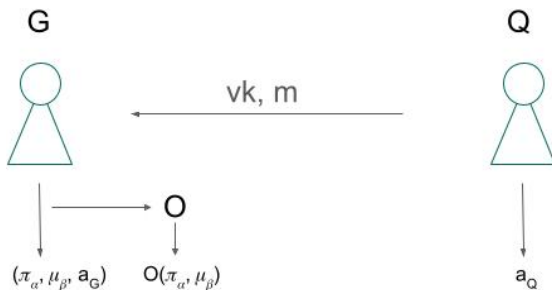
Properties Required:

- 1 Correctness.
- 2 Unforgeability: without sk , hard to forge signature on a new message.

Example 2: Signature-Checking R3PO

Interaction:

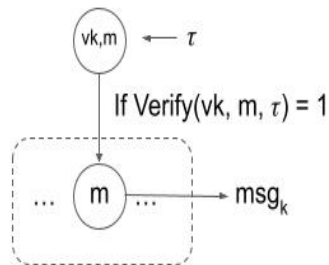
- Q sends vk, m to G .



Example 2: Signature-Checking R3PO

Reactive Program:

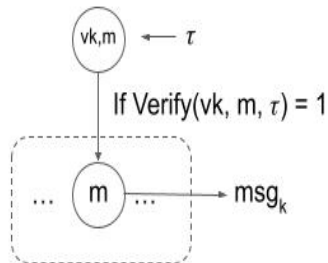
$$\pi(\text{st}_{\text{vk},m}^1, \tau) = \begin{cases} \text{st}_m^2, & \text{if } \text{verify}(\text{vk}, m, \tau) = 1 \\ \perp, & \text{else} \end{cases}$$



Example 2: Signature-Checking R3PO

Reactive Program:

$$\pi(\text{st}_{\text{vk},m}^1, \tau) = \begin{cases} \text{st}_m^2, & \text{if } \text{verify}(\text{vk}, m, \tau) = 1 \\ \perp, & \text{else} \end{cases}$$



Theorem 2 (Informally)

If the DDH assumption holds, there exists a Signature scheme and a R3PO for Signature-Checking.

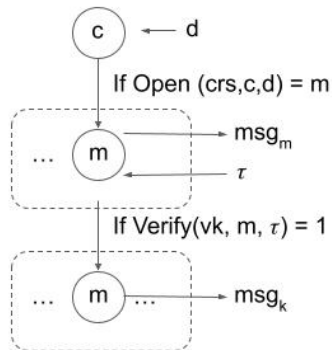
Towards R3PO of Larger Reactive Programs

Can we combine R3PO for Commitment Opening and Signature Checking?

Reactive Program:

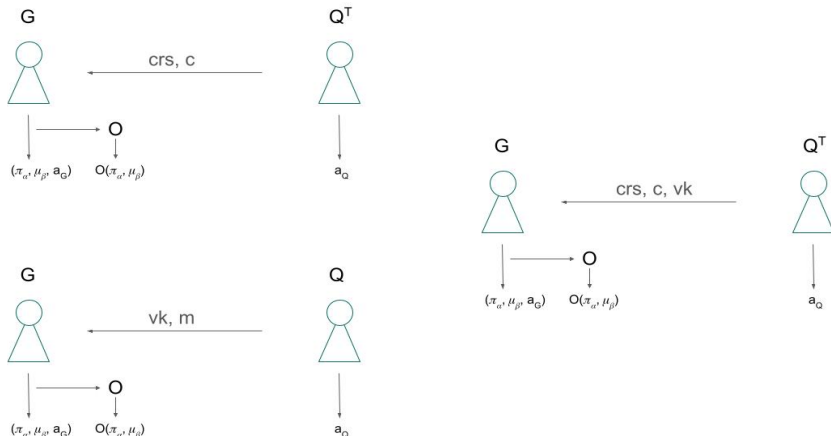
$$\pi(\text{st}_c^1, d) = \begin{cases} \text{st}_m^2, & \text{if } \text{open}(\text{crs}, c, d) = m \\ \perp, & \text{else} \end{cases}$$

$$\pi(\text{st}_m^2, \tau) = \begin{cases} \text{st}_m^3, & \text{if } \text{verify}(\text{vk}, m, \tau) = 1 \\ \perp, & \text{else} \end{cases}$$



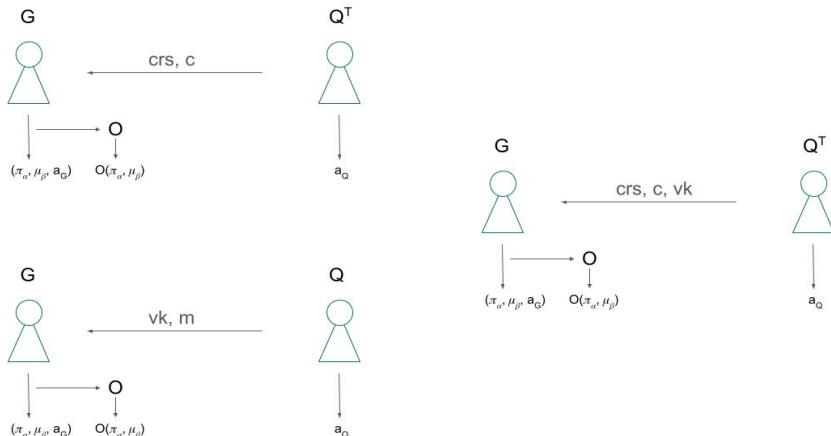
Towards R3PO of Larger Reactive Programs

Can we combine R3PO for Commitment Opening and Signature Checking?



Towards R3PO of Larger Reactive Programs

Can we combine R3PO for Commitment Opening and Signature Checking?

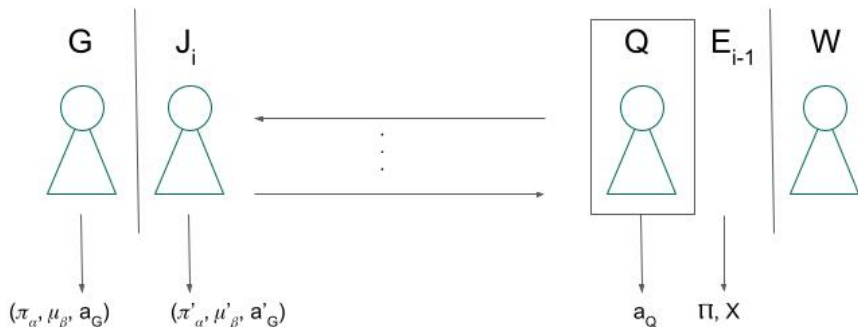


Need to be careful in handling the interaction!

Decomposition Property

We say that a generator class \mathcal{G} **decomposes** to a generator class \mathcal{G}_i (at partition i) if the following bi-simulations are indistinguishable.

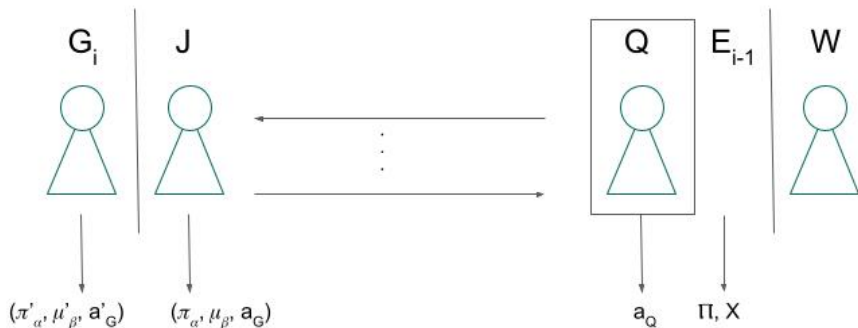
- In the interaction between $G \in \mathcal{G}$ and $Q \in \mathcal{Q}$, for all $(i-1)$ -partial reach extractors E_{i-1} , there exists J_i, W s.t. J_i outputs $(\pi'_\alpha, \mu'_\beta, a'_G)$.



Decomposition Property

We say that a generator class \mathcal{G} **decomposes** to a generator class \mathcal{G}_i (at partition i) if the following bi-simulations are indistinguishable.

- In the interaction between $G_i \in \mathcal{G}_i$ and $Q|E_{i-1}|W \in \mathcal{Q}$, there exists J that outputs $(\pi_\alpha, \mu_\beta, a_G)$.



Composition Theorem

Theorem 3 (Informally)

Let $\mathcal{G}_1, \dots, \mathcal{G}_n$ be generator classes with R3PO schemes $\mathcal{O}_1, \dots, \mathcal{O}_n$. If a generator class \mathcal{G} decomposes to \mathcal{G}_i at each partition $i \in [n]$, then there exists a R3PO scheme for \mathcal{G} .

The proof uses garbled-circuit chaining.

Theorem 3 (Informally)

Let $\mathcal{G}_1, \dots, \mathcal{G}_n$ be generator classes with R3PO schemes $\mathcal{O}_1, \dots, \mathcal{O}_n$. If a generator class \mathcal{G} decomposes to \mathcal{G}_i at each partition $i \in [n]$, then there exists a R3PO scheme for \mathcal{G} .

Corollary:

If there exists a R3PO for commitment-opening, then:

- there exists a R3PO for sequence of commitment-openings
- there exists a 2-round MPC protocol secure against semi-honest dishonest majority corruption [BL17, GS17]

Theorem 3 (Informally)

Let $\mathcal{G}_1, \dots, \mathcal{G}_n$ be generator classes with R3PO schemes $\mathcal{O}_1, \dots, \mathcal{O}_n$. If a generator class \mathcal{G} decomposes to \mathcal{G}_i at each partition $i \in [n]$, then there exists a R3PO scheme for \mathcal{G} .

Corollary:

If there exists a R3PO for signature-checking, then:

- there exists a R3PO for sequence of signature-checkings
- there exists an adaptive-secure IBE scheme [DG17a, DG17b]

Composition Theorem

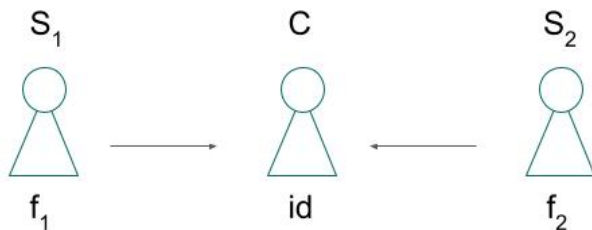
Theorem 3 (Informally)

Let $\mathcal{G}_1, \dots, \mathcal{G}_n$ be generator classes with R3PO schemes $\mathcal{O}_1, \dots, \mathcal{O}_n$. If a generator class \mathcal{G} decomposes to \mathcal{G}_i at each partition $i \in [n]$, then there exists a R3PO scheme for \mathcal{G} .

Corollary:

If there exists a R3PO for commitment-opening and a R3PO for signature-checking, then:

- there exists a R3PO for commitment-opening followed by signature checking.
- if there exists an ABE scheme, there exists a “private” MA-ABE scheme (our work).

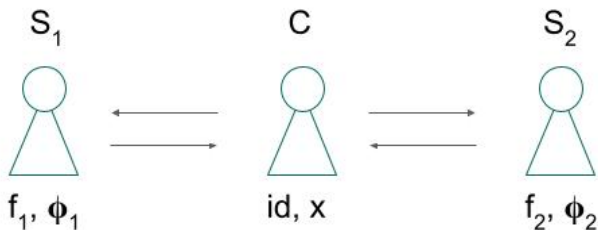


Global ID Model [Cha07, LW10]:

- Each client has a global id
- Only interaction: servers send credentials for id to a client
- Current results rely on the Random Oracle Model: E.g., [DKW20] for DNF formulae under the LWE assumption.

Private MA-ABE: A client can privately decide on the attributes it wants to acquire, as long as it conforms to the servers' policy. Client can send a message to each server first.

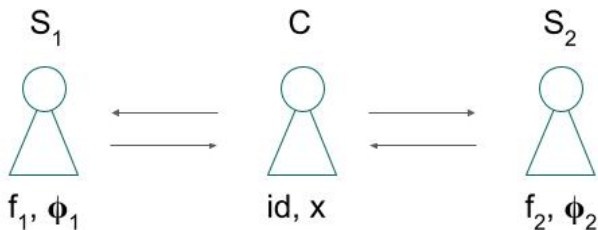
Attribute Verification



2-round protocol for Attribute Verification:

- Decentralized setup of Servers: publish global public keys.
- Round 1: C sends a request to S_1 and S_2 .
- Round 2: Servers S_1 and S_2 send response to C .

Attribute Verification

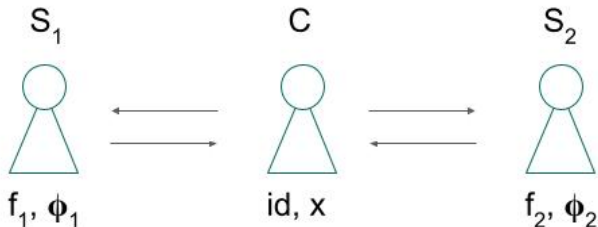


Completeness:

if $\Phi_1(id, x) = 1$ and $\Phi_2(id, x) = 1$, then C gets $f_1(id, x)$, $f_2(id, x)$.

Hiding: Server S_b learns nothing about (id, x) and Φ_{1-b} .

Attribute Verification



Solution using R3PO:

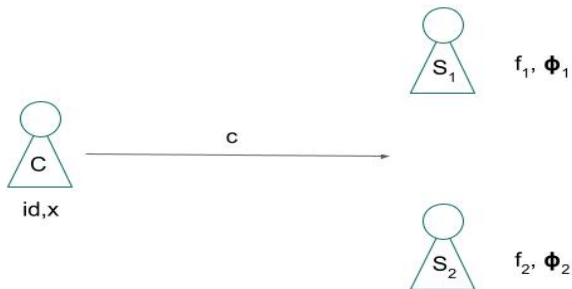
Use commitment to hide (id, x) .

Use signatures to give proof of verification.

Attribute Verification

Round 1:

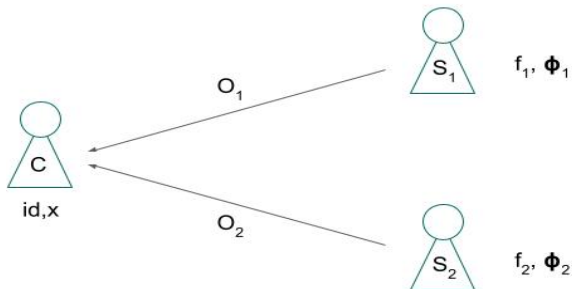
- Client C computes $(c, d) \leftarrow \text{commit}(\text{crs}, \text{id})$ and sends c to servers S_1, S_2 .



Attribute Verification

Round 2:

- Server S_1 sends \mathcal{O}_1 to Client C .
- Server S_2 sends \mathcal{O}_2 to Client C .



Attribute Verification

Round 2:

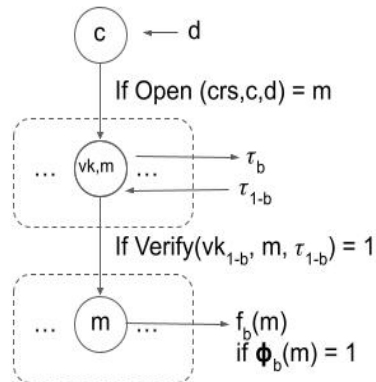
- Server S_1 sends \mathcal{O}_1 to Client C .
- Server S_2 sends \mathcal{O}_2 to Client C .

where, each \mathcal{O}_i is R3PO of program with:
transition function π :

$$\begin{aligned}\pi(\text{st}_c^1, d) &= \text{st}_m^2, \text{ if } \text{open}(\text{crs}, c, d) = m \\ \pi(\text{st}_m^2, \tau) &= \text{st}_m^3, \text{ if } \text{verify}(\text{vk}_{1-b}, m, \tau_{1-b}) = 1\end{aligned}$$

message function μ :

$$\begin{aligned}\mu_{\text{sk}_b, f_b, \Phi_b}(\text{st}_m^3) &= \text{sign}(\text{sk}_b, m) \\ \mu_{\text{sk}_b, f_b, \Phi_b}(\text{st}_m^3) &= f_b(m), \text{ if } \Phi_b(m) = 1\end{aligned}$$



Theorem 4

If there exists an R3PO for commitment-opening and signature-checking, then there exists a 2-round Protocol for Attribute Verification.

Theorem 4

If there exists an R3PO for commitment-opening and signature-checking, then there exists a 2-round Protocol for Attribute Verification.

Corollary: Given the following primitives:

- a CP-ABE scheme for general policies
- R3PO for commitment-opening and signature-checking

there exists a Private MA-ABE scheme for general policies.

In Conclusion

- R3PO: Obfuscation of interactively sampled programs
- A library of R3PO instantiations from standard assumptions:
 - Commitment-Opening
 - Signature-Checking
 - Can optionally restrict to a message prefix.
 - Hash-Checking
- A composition theorem to build R3PO for larger program classes.
 - Encapsulates Garbled Circuit Chaining technique
- As an application, we construct Private MA-ABE
- **Open Directions:** More applications, capturing more constructions (e.g., Garbled RAM), adding more features (e.g., blindness)

Thank You

References I

- [ABG⁺13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry, Differing-inputs obfuscation and applications, Cryptology ePrint Archive, Paper 2013/689, 2013, <https://eprint.iacr.org/2013/689>.
- [BBC⁺13] Boaz Barak, Nir Bitansky, Ran Canetti, Yael Tauman Kalai, Omer Paneth, and Amit Sahai, Obfuscation for evasive functions, Cryptology ePrint Archive, Paper 2013/668, 2013, <https://eprint.iacr.org/2013/668>.
- [BCTKP14] Nir Bitansky, Ran Canetti, Yael Tauman-Kalai, and Omer Paneth, On virtual grey box obfuscation for general circuits, Cryptology ePrint Archive, Paper 2014/554, 2014, <https://eprint.iacr.org/2014/554>.
- [BGI⁺01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang, On the (im)possibility of obfuscating programs, Cryptology ePrint Archive, Paper 2001/069, 2001, <https://eprint.iacr.org/2001/069>.
- [BL17] Fabrice Benhamouda and Huijia Lin, k-round mpc from k-round ot via garbled interactive circuits, Cryptology ePrint Archive, Paper 2017/1125, 2017, <https://eprint.iacr.org/2017/1125>.
- [CDG⁺17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou, Laconic oblivious transfer and its applications, Cryptology ePrint Archive, Paper 2017/491, 2017, <https://eprint.iacr.org/2017/491>.
- [Cha07] Melissa Chase, Multi-authority attribute based encryption, Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings, Lecture Notes in Computer Science, vol. 4392, Springer, 2007, pp. 515–534.
- [DG17a] Nico Döttling and Sanjam Garg, From selective ibe to full ibe and selective hibe, Cryptology ePrint Archive, Paper 2017/957, 2017, <https://eprint.iacr.org/2017/957>.
- [DG17b] _____, Identity-based encryption from the diffie-hellman assumption, Cryptology ePrint Archive, Paper 2017/543, 2017, <https://eprint.iacr.org/2017/543>.
- [DKW20] Pratish Datta, Ilan Komargodski, and Brent Waters, Decentralized multi-authority abe for dnfs from lwe, Cryptology ePrint Archive, Paper 2020/1386, 2020, <https://eprint.iacr.org/2020/1386>.
- [GS17] Sanjam Garg and Akshayaram Srinivasan, Two-round multiparty secure computation from minimal assumptions, Cryptology ePrint Archive, Paper 2017/1156, 2017, <https://eprint.iacr.org/2017/1156>.

- [HRsV07] Susan Hohenberger, Guy N. Rothblum, abhi shelat, and Vinod Vaikuntanathan, Securely obfuscating re-encryption, Theory of Cryptography (Berlin, Heidelberg) (Salil P. Vadhan, ed.), Springer Berlin Heidelberg, 2007, pp. 233–252.
- [IPS14] Yuval Ishai, Omkant Pandey, and Amit Sahai, Public-coin differing-inputs obfuscation and its applications, Cryptology ePrint Archive, Paper 2014/942, 2014, <https://eprint.iacr.org/2014/942>.
- [JLS20] Aayush Jain, Huijia Lin, and Amit Sahai, Indistinguishability obfuscation from well-founded assumptions, Cryptology ePrint Archive, Paper 2020/1003, 2020, <https://eprint.iacr.org/2020/1003>.
- [LW10] Allison Lewko and Brent Waters, Decentralizing attribute-based encryption, Cryptology ePrint Archive, Paper 2010/351, 2010, <https://eprint.iacr.org/2010/351>.
- [WZ17] Daniel Wichs and Giorgos Zirdelis, Obfuscating compute-and-compare programs under lwe, Cryptology ePrint Archive, Paper 2017/276, 2017, <https://eprint.iacr.org/2017/276>.