Rational Protocol Design

Motivation

- Secure MPC protocols offer security against two classes of adversaries; *semi-honest* adversaries and *malicious* adversaries.
- *Semi-Honest* adversaries are too optimistic to assume; *malicious* adversaries are too pessimistic.
- We require a middle-ground; security against adversaries that deviate only if incentivised to do so.

Rational Cryptography

- Analysing security of cryptographic protocols assuming that the parties involved in the protocol are rational.
- **Rational Secret Sharing**: parties are given shares to a secret by a trusted third party and they have to reconstruct the secret.
- **Rational Multiparty Computation**: The party P_i has a value x_i such that the parties want to securely compute f(x_1,.....,x_n) such that the parties involved in the computation are rational.
- The parties involved in this protocol have incentive to learn the output of the protocol and stop other parties from learning the output.

- Rational Protocol Design by Garay et. al. \cite{} present a new framework, where the parties themselves are assumed to be “honest”, and the adversary corrupting parties is assumed to be incentive-driven.

Overview

- Propose a model for incentive-driven attacks; a two-party game between the protocol designer D and adversary A that corrupts a subset of parties in the protocol and deviates from protocol execution.
- D and A are unbounded; can be modelled as a Zero Sum Game with perfect information and observable action (Stackleberg Game)
- Modelling the utility of attacker: Execution in the real protocol can be simulated by an ideal adversary interacting with a defective ideal functionality <F>; utility of the attacker proportional to the probability of interacting with the defective functionality that lead to “security breaches”

Contd.

- Defective Ideal functionality <F>
  - (query, \stackrel{x_I}{\rightarrow}) : The ideal functionality returns the output $f(\stackrel{x_I}{\rightarrow}, \stackrel{x_{-I}}{\rightarrow})$ to the ideal adversary.
  - (out,y) : Upon receiving this message, the ideal functionality replaces the output of the function by y. Let this event be called *Breaking Correctness* event.
  - (inp, \stackrel{x_I}{\rightarrow}) : upon the ideal adversary sending this query, it get the value $f(\stackrel{x'_I}{\rightarrow}, \stackrel{x_{-I}}{\rightarrow})$ from the functionality, where $x'_I \neq x_I$. Let this event be *Breaking Privacy* event.

Utility of Attacker

- Corrupting subset of parties: The attacker incurs a cost of γ_I when it corrupts the subset I of parties. A simpler model is considered where cost of corrupting one party = $\gamma_{$}$
- Breaking Correctness: The attacker gains a utility of γ_c if it triggers the *Breaking Correctness* event in the defective ideal functionality
- Breaking Privacy: The attacker gains a utility of γ_c if it triggers the *Breaking Privacy* event in the defective ideal functionality

Results

- Secure Function Evaluation protocol (Attack Payoff Secure) for computing arbitrary functionality assuming cost of corruption >> breaking privacy utility.
- Conversely, there exist function f such that it can’t be securely evaluated assuming cost of corruption < breaking privacy/correctness utility.
- In 2-party setting, the above impossibility result is sidestepped; the authors propose an attack payoff optimal protocol to evaluate arbitrary function f

Two party Attack Game G_M

- Define the game M = (F,<F>,v), where F is the functionality, <F> is the defective ideal functionality, and v is the utility function.
- First move by the protocol designer D. Chooses a protocol to compute functionality F from the set of n-party polynomial time computable protocols. Sends the protocol to A
- The attacker A chooses an ITM \Adv to attack the protocol and the utility of the attacker is defined proportional to the probability that it triggers *corrupting party/breaking correctness/breaking privacy* event in the ideal model.
- The notion of equilibrium used is \epsilon-subgame perfect equilibrium

Contd.

v_{<F>,S,Z} -> Random variable denoting the utility of ideal attacker S with access to functionality <F> in the environment Z

U_I^{<F>}(S,Z) = E(v_{<F>,S,Z})

Expected payoff of attacker A in the real world -> U_i^{\Pi,<F>}(A,Z) = inf_{S} (U_I^{<F>}(S,Z))

Why minimum?

Maximal payoff of attacker A is the quantity we consider as the utility of A. It is defined as:

U_i^{\Pi,<F>}(A) = max_{Z} (U_i^{\Pi,<F>}(A,Z))

Security Definitions

An ITM A is the M-maximising adversary for protocol Π if

U^{\Pi,<F>}(A) \compindis max_{A'}U^{\Pi,<F>}(A') =: U^{\Pi,<F>}

Protocol Π is attack payoff optimal in M if for any other protocol Π',

U^{\Pi,<F>} <= U^{\Pi',<F>} + negl

Protocol Π is attack payoff secure in M if for any other protocol Π',

U^{\Pi,<F>} <= U^{\phi^{F},<F>} + negl

Is there a difference between attack-payoff optimal and attack-payoff secure?

Interesting Intermediate Results

Theorem 1: A protocol \Pi is attack-payoff optimal in M iff the strategy profile (\Pi,A) is \epsilon-subgame-perfect equilibrium of M

Theorem 2 (Universal Composibilty) Any sub-protocol can be replaced with the corresponding ideal functionality without affecting the utility of **M-maximising adversary**

Secure Function Evaluation for $\gamma_p < n/2 \ gamma_{$} $.

Idea: Take a protocol that is secure in the presence of honest majority. If the adversary corrupts a minority of parties, then the protocol is secure. If it corrupts a majority of parties, the cost of corruption exceeds the *Breaking Privacy* utility earned by the attacker.

The protocol should have the following features:

1. The parties should commit to their inputs before the protocol, thus limiting the privacy breaking queries the attacker can ask.
2. If a party refuses to reveal its shares or aborts during protocol execution, the protocol should restart with the other parties inputs (See Guaranteed Output Delivery)

Protocol $\Pi^{f}_{\text{St-SFE}}$

PHASE 1 (INPUT COMMITMENT): The following steps are executed. Initially $D := \emptyset$:

1.1 Every party $p_i \in \mathcal{P}$ computes a commitment on his input and broadcasts it. Denote this value by com_i and the corresponding decommitment information (known exclusively to p_i) by dec_i. Any party that does not broadcast a commitment is added to D.

1.2 Every p_j signs (using his private signing key sk_j) all the commitments $\mathsf{com}_{\ell_1}, \ldots, \mathsf{com}_{\ell_{|\mathcal{P}\setminus D|}}$ broadcast by the parties in $\mathcal{P} \setminus D = \{p_{\ell_1}, \ldots, p_{\ell_{|\mathcal{P}\setminus D|}}\}$, and broadcasts these signatures. If some p_j broadcasts an inconsistent message or an invalid signature, then $D := D \cup \{p_j\}$.

PHASE 2 (COMPUTATION): Let $\vec{S}|_{\mathcal{P}\setminus D}$ denote the vector of all commitments and signatures from parties in $\mathcal{P}\setminus D$. Using an SFE protocol which is secure with identifiable abort for arbitrarily many corruptions (e.g., Π_{CFGN}), the parties in $\mathcal{P}\setminus D = \{p_{\ell_1}, \ldots, p_{\ell_{|\mathcal{P}\setminus D|}}\}$ evaluate the functionality $\mathcal{F}^{f}_{\text{COM-SFE}}(\mathcal{P}\setminus D, \lfloor \frac{n}{2} \rfloor - |D|, (\mathsf{pk}_{\ell_1}, \ldots, \mathsf{pk}_{\ell_{|\mathcal{P}\setminus D|}}))$ on input $\vec{S}|_{\mathcal{P}\setminus D}$. If the evaluation outputs $(\mathtt{detect}, p_i)$ or aborts with p_i then set $D := D \cup \{p_i\}$ and repeat Phase 2 in this updated setting.

PHASE 3 (OUTPUT): Let $s_{\ell_1}, \ldots, s_{\ell_{|\mathcal{P}\setminus D|}}$ be the shares output in Phase 2.

3.1 Every party broadcasts his share s_i along with the corresponding signature.

3.2 If at least $\lfloor \frac{n}{2} \rfloor - |D| + 1$ announced shares with valid signatures, then interpolate the corresponding polynomial and output the shared value. Otherwise, let D' denote the set of parties that announced no share or an invalid signature. Set $D := D \cup D'$ and repeat Phase 2 in this updated setting.

Functionality $\mathcal{F}^{f}_{\text{COM-SFE}}\left(\mathcal{P}, t, (\mathsf{pk}_1, \ldots, \mathsf{pk}_{|\mathcal{P}|})\right)$

The functionality is parametrized by a function $f : \mathbb{F}^{|\mathcal{P}|} \to \mathbb{F}$, a player set $\mathcal{P}$, a threshold $t \leq |\mathcal{P}|$, a non-interactive commitment scheme, a signature scheme, and a vector of public (verification) keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_{|\mathcal{P}|})$. $\mathcal{F}_{\text{COM-SFE}}$ proceeds in rounds/steps as described below, keeping a set D of detected parties which is initialized to $D := \emptyset$.

1. Every $p_i \in \mathcal{P}$ hands $\mathcal{F}_{\text{COM-SFE}}$ a vector $(\mathsf{com}_{i,1}, \ldots, \mathsf{com}_{i,|\mathcal{P}|})$ of commitments along with signatures on each $\mathsf{com}_{i,j}$ matching each of the keys $\mathsf{pk}_k, k \in \{1, \ldots, |\mathcal{P}|\}$; furthermore, every $p_i \in \mathcal{P}$ hands $\mathcal{F}_{\text{COM-SFE}}$ a decommitment dec_i. If some p_i does not send $|\mathcal{P}|$ commitments along with the all $|\mathcal{P}|^2$ corresponding valid signatures then output $(\texttt{detect}, p_i)$ and halt (if there are many such p_i's output the one with the smallest index).[a] Denote by $\vec{S}$ the set of all valid signed commitments.
2. $\mathcal{F}_{\text{COM-SFE}}$ computes the following set for each $p_i \in \mathcal{P}$:

$$C_i := \{\mathsf{com}_{j,i} \mid \mathsf{com}_{j,i} \in \vec{S} \text{ has valid signatures for every } \mathsf{pk}_k, k \in [|\mathcal{P}|]\}$$

   If for some $p_i : |C_i| \neq 1$ then set $(\texttt{detect}, p_i)$ and halt (if there is many such p_i's output the one with the smallest index). Otherwise, for each $p_i \in \mathcal{P}$ denote by com_i the unique element of C_i.
3. For each p_i use dec_i to open com_i. If opening fails then issue $(\texttt{detect}, p_i)$ and halt. Otherwise, let x_i denote the opened value.
4. Compute $y := f(x_1, \ldots, x_n)$. Choose a polynomial $g \in \mathbb{F}[X]$ of degree at most t uniformly at random so that $g(0) = y$; for each $p_j \in \mathcal{P}$ set $s_j := g(j)$.
5. Use the key generation algorithm for generating a (fresh) signing/verification key-pair $(\mathsf{sk}, \mathsf{pk})$; for each $p_j \in \mathcal{P}$ compute a signature σ_j on s_j using key sk.
6. For each $p_j \in \mathcal{P}$ output $(s_j, \sigma_j, \mathsf{pk})$ to p_j.

[a] The signatures are assumed to have a unique party ID and message ID, which are checked to determine their validity.

Impossibility result when *Breaking Privacy/Correctness* cost is significantly greater than cost of corruption

Fact 1 ([Kat07, Theorem 1]). *There is a finite, deterministic functionality* $\mathcal{F}_{\text{K07}}$ *for which there is no polynomial-round protocol* Π *that simultaneously satisfies the following two properties: (1)* Π *securely realizes* $\mathcal{F}_{\text{K07}}$ *in the presence of* t *corrupted parties, and (2)* Π *is* $(n-t)$*-private.*

Theorem 8. *Let* $t \geq n/2$. *If* $\gamma_{\mathsf{p}} > \gamma_{\$} t$ *and* $\gamma_{\mathsf{c}} > \gamma_{\$}(n-t)$, *then there exists no (polynomial-round) attack-payoff secure protocol* Π *in the attack model* $(\mathcal{F}_{\text{K07}}, \langle \mathcal{F}_{\text{K07}} \rangle, v^{\vec{\gamma}})$.

Positive Result for an Impossibility Case

Theorem 10. *Let* $\mathcal{M} = (\mathcal{F}_{\text{SFE}}, \langle \mathcal{F}_{\text{SFE}} \rangle, v^{\vec{\gamma}})$ *be an attack-model where* $\min\{\gamma_{\mathsf{p}}, \gamma_{\mathsf{c}}\} > \gamma_{\$}$. *Then for any static adversary attacking* $\Pi_{\mathtt{Op-SFE}}$ *in* $\mathcal{M}$, $\hat{U}^{\Pi_{\mathtt{Op-SFE}}, \langle \mathcal{F}_{\text{SFE}} \rangle} \overset{\text{negl}}{\leq} \frac{\min\{\gamma_{\mathsf{p}}, \gamma_{\mathsf{c}}\}}{2} - \gamma_{\$}$.