# Towards a Game Theoretic View of Secure computation

# Objective: (Security and Correctness)

- We relate the two formalisms.
- We demonstrate how Game Theoretic concepts and formalisms can be used to capture cryptographic notions of security of protocols.
- In the setting of two party protocols and fail-stop adversaries, we show how the traditional notions of secrecy and correctness of protocols can be captured as properties of Nash equilibria in games for rational players.
- We first show Game theoretic notions of secrecy and correctness that are equivalent, respectively, to the standard cryptographic notions of secret and correct evaluation of deterministic functions in fail-stop setting.

# Objective: (Fairness)

- We formulate a natural Game theoretic notion of fairness and observe that it is strictly weaker than existing cryptographic notions of fair two party function evaluation.
- Next, we formulate new cryptographic notions of fairness that are equivalent to this Game theoretic notion, and a simulation based notion of fairness that implies the above three.

# Basic Idea

- We translate a given protocol into a set of games, in such a way that the protocol satisfies the cryptographic property in question if and only if a certain pair of strategies are in a (computational) Nash equilibrium in each one of the games.
- Precisely, given a protocol, we consider the game where in each step the relevant party can decide to either continue running the protocol as prescribed, or alternatively abort the execution. We then ask whether the pair of strategies that instruct the players to continue the protocol to completion is in a (computational) Nash equilibrium. Each cryptographic property is then captured by an appropriate set of utilities and input distributions (namely, distributions over the types)

# Secrecy

- A given protocol is secret if and only if the strategy that never aborts the protocol is in a computational Nash equilibrium with respect to the following set of utilities and distributions over the types.
- How do we define secrecy here?

For each pair of values in the domain, we define a distribution that chooses an input for one party at random from the pair. The party gets low payoff if the two values lead to the same output value and yet the other party managed to guess which of the two inputs was used.

# Correctness

- A protocol correctly computes a deterministic function if and only if the strategy that never aborts the protocol is in a computational Nash equilibrium with respect to the set of utilities
- The utilities are in such a way that the parties get high payoff only if they output the correct function value on the given inputs (types), or abort before the protocol starts; in addition, the players get no payoff for incorrect output.

# Fairness

- Basic setting: Two parties interact by exchanging messages in order to valuate a function f on their inputs.
- The only allowed deviation from the protocol is abortion, in which event both parties learn that the protocol was aborted.
- Basically, a protocol in this model should specify, in addition to the next message to be sent, also a prediction of the output value in case the execution is aborted. (fair exchange function, where the output of each party is the input of the other)

# Current notion of fairness

- Current notions of fairness for two party protocols require there to be a point in the computation where both parties move from a state of no knowledge of the output to a full knowledge of it. [Strong notion]

# Our notion

- We relax our notion of fairness.
- **Gradual Release**
- We let each party has, in addition to its own input, some additional information on the input of the other party. (perhaps two possible values of input)
- For each **valid** quadruple, an input for one party is chosen at random from the first two values and an input for other party is chosen at random from the other two values.
- When the party aborts the protocol, each party predicts its output.
- Utility is defined in a way that, if the party predicts correctly (incorrectly) and the other one does not, then it gets payoff +1 (-1). Else, 0.

# Three different cryptographic notions of fairness

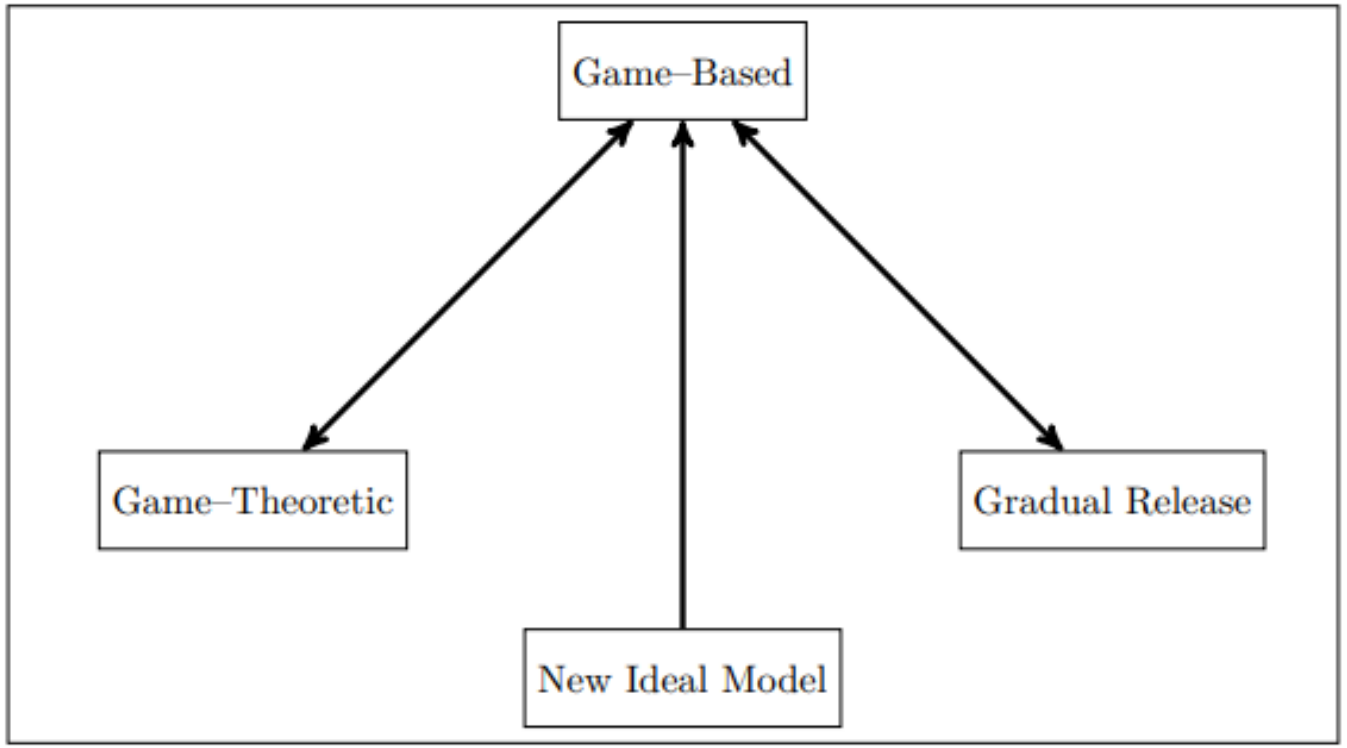


Figure 1: Our four notions of fairness and their relationships

# “game-based” notion of fairness

- limits the gain of an arbitrary fail-stop adversary in a game.
- closely mimics the above Game Theoretic interaction except here the adversary is arbitrary, rather than rational.
- We show that the two notions are equivalent.

# Concept of limited gradual release

- The probability of any party to predict its output increases only by a negligible amount.
- We show that a protocol is fair if and only if it satisfies the limited gradual release property.

# Ideal-model-based notion of fairness

- Allows for limited gradual release of secrets.
- The ideal functionality accepts a “sampling algorithm” M from the ideal-model adversary.
- The functionality then obtains the inputs from the parties and runs M on these inputs, and obtains from M the outputs that should be given to the two parties.
- The functionality then makes the respective outputs available to the two parties.
- We require M to be both “fair” and “correct” i.e. both parties get correct output with roughly equal probability.

# On the Definitional Choices

- Why use plain Nash equilibria to exhibit correspondence between cryptographic notions and Game Theoretic ones?
- Why not, for instance, stronger notions such as Dominant Strategy, Survival Under Iterated Deletions or Subgame Perfect equilibria?
- It turns out that in our setting of two-party computation with fail-stop faults, Nash equilibria do seem to naturally correspond to cryptographic secure protocols. In particular, in the fail-stop case any Nash equilibrium is sub-game perfect, or in other words empty threats do not hold.

# Organization

- the Game Theoretic notion
- the equivalent cryptographic definition
- a new simulation-based definition
- the limited gradual release property and its relation to fairness
- the study of the fairness definition

# A positive result!

- Impossibility results, of Cleve and Asharov-Lindell, hold even with respect to this weaker notion, as long as both parties are required to receive an output.
- Our notion is meaningful even in the case where parties are not guaranteed to always learn the correct output when played honestly.
- In cases where correctness holds with probability between 0 and ½, our simulation-based notion of fairness is achievable with no set-up or trusted third-parties.

# Our work, Related Work, Motivation

- Cleve's impossibility result extension
- We show fairness is impossible to achieve even if correctness holds with probability <1 (>0.5)
- We overcome this impossibility by relaxing the requirements and show that fairness can be achieved for protocols with correctness ½.
- Recent work, different approach to overcome impossibility [they relax utility instead of correctness]

# Review of some cryptographic definitions

**Negligible functions and indistinguishability.** A function $\mu(\cdot)$ is negligible if for every polynomial $p(\cdot)$ there exists a value $N$ such that for all $n > N$ it holds that $\mu(n) < \frac{1}{p(n)}$. Let $X = \{X(a,n)\}_{n\in N, a\in\{0,1\}^*}$ and $Y = \{Y(a,n)\}_{n\in N, a\in\{0,1\}^*}$ be distribution ensembles. Then, we say that $X$ and $Y$ are computationally indistinguishable, denoted $X \stackrel{c}{\equiv} Y$, if for every non-uniform probabilistic polynomial time (PPT) distinguisher $D$ there exists a negligible function $\mu(\cdot)$ such that for all sufficiently long $a \in \{0,1\}^*$,

$$|\Pr[D(X(a,n)) = 1] - \Pr[D(Y(a,n)) = 1]| < \mu(n).$$

# Review of some cryptographic definitions

- Protocol: In our setting, we study two party protocols, modelled as a pair of interacting Turing machines.
- We formulate both the cryptographic and the Game Theoretic concepts in terms of two-party protocols.
- We restrict attention to PPT machines.
- [Here, to simplify the analysis, we consider machines that are polynomial in a globally known security parameter, rather than in the length of their inputs.]

# Review of some cryptographic definitions

**Two-Party functions.** In general, a *two-party function* is a probability distribution over functions $f : \{0,1\}^* \times \{0,1\}^* \times \mathbf{N} \to \{0,1\}^* \times \{0,1\}^*$. Here the first (second) input and output represent the input and output of the first (second) party, and the third input is taken to be the security parameter. In this work we consider the restricted model of deterministic functions. We say that a function is *efficiently invertible* if, for $i = 0, 1$, given $1^n$, an input value $x_i$ and an output value $y$, it is possible to compute in PPT a value $x_{1-i}$ such that $f(x_0, x_1, n) = y$.

# Review of some cryptographic definitions

The Fail-stop setting:

- We consider two-party interaction in the presence of fail-stop faults.
- Both parties follow the protocol specification exactly, with the exception that any one of the parties may, at any time during the computation, decide to stop, or abort the computation.
- The abortion operation is explicit and public.
- Note: This modelling of abortion as a public operation is easily justified in a communication setting with reasonable timeouts on the communication delays. Protocols in this model should specify the output of a party in case of early abortion of the protocol. We assume that this output has a format that distinguishes this output from output that does not result from early abortion.

# Fail-stop Adversary

- Fail-stop adversaries do not change their initial input for the execution, yet, they may arbitrarily decide on their output.
- The Game Theoretic and the cryptographic approaches differ in the specifics of the abortion step.
- In both cases we assume that the abortion operation is explicit and public.
- As soon as one party decides to abort, the other party receives an explicit notification of this fact and can act accordingly. (in contrast to the setting where one party decides to abort while the other party keeps waiting indefinitely to the next incoming message.)

# Cryptographic Security

# Some definitions

- View:

By definition [10], the **View** of the $i$th party ($i \in \{0,1\}$) during an execution of $\pi$ on $(x_0, x_1)$ is denoted $\mathbf{View}_{\pi,i}(x_0, x_1, n)$ and equals $(x_i, r^i, m_1^i, ..., m_t^i)$, where $r^i$ equals the contents of the $i$th party's *internal* random tape, and $m_j^i$ represents the $j$th message that it received.

# Privacy and some Intuitive idea:

- We introduce a definition of private computation.
- Intuitively, it means that no party (that follows the protocol) should be able to distinguish any two executions when using the same inputs and seeing the same outputs. This holds even for the case that the other party uses different inputs.

# Formal definition

**Definition 2.1 (privacy)** *Let $f$ and $\pi$ be as above. We say that $\pi$* privately computes *$f$ if the following holds:*

1. *For every non-uniform* PPT *adversary $\mathcal{A}$ that controls party $P_0$*

   $$\left\{\mathbf{View}_{\pi,\mathcal{A}(z),0}(x_0,x_1,n)\right\}_{x_0,x_1,x_1',y,z\in\{0,1\}^*,n\in\mathbb{N}} \stackrel{c}{\equiv} \left\{\mathbf{View}_{\pi,\mathcal{A}(z),0}(x_0,x_1',n)\right\}_{x_0,x_1,x_1',z\in\{0,1\}^*,n\in\mathbb{N}}$$

   *where $|x_0|=|x_1|=|x_1'|$ and $f(x_0,x_1)=f(x_0,x_1')$.*

2. *For every non-uniform* PPT *adversary $\mathcal{A}$ that controls party $P_1$*

   $$\left\{\mathbf{View}_{\pi,\mathcal{A}(z),1}(x_0,x_1,n)\right\}_{x_0,x_0',x_1,z\in\{0,1\}^*,n\in\mathbb{N}} \stackrel{c}{\equiv} \left\{\mathbf{View}_{\pi,\mathcal{A}(z),1}(x_0',x_1,n)\right\}_{x_0,x_0',x_1,z\in\{0,1\}^*,n\in\mathbb{N}}$$

   *where $|x_0|=|x_0'|=|x_1|$ and $f(x_0,x_1)=f(x_0',x_1)$.*

# Correctness (informal)

- We distinguish between output that corresponds to successful termination of the protocol and output generated as a result of an abort message.
- The second output starts with a $\perp$ sign.
- The correctness requirement only applies to the first type of output.
- For the fail-stop setting, it holds that privacy and correctness imply simulation-based security with abort.

# Formal definition

**Definition 2.2 (correctness)** *Let $f$ and $\pi$ be as above. We say that $\pi$* correctly computes *$f$ if for all sufficiently large inputs $x_0$ and $x_1$ such that $|x_0| = |x_1| = n$, we have:*

$$\Pr[\mathbf{Output}_{\pi,i} \in \{\perp \circ \{0,1\}^*, f(x_0, x_1)\}] \geq 1 - \mu(n)$$

*where $\mathbf{Output}_{\pi,i} \neq \perp$ denotes the output returned by $P_i$ upon the completion of $\pi$ whenever the strategy of the parties is* `continue`*, and $\mu$ is a negligible function.*

# Review of some concepts and Game theoretic definitions

- Here, we extend some concepts to put then on equal footing as cryptographic concepts. (introducing asymptotic, computationally bounded players, and negligible error probabilities.)
- We consider the case of two-player games.
- Traditionally a 2-player (normal form, full information) game $\Gamma = (\{A_0, A_1\}, \{u_0, u_1\})$ is determined by specifying, for each player $P_i$, a set $A_i$ of possible actions and a utility function. $u_i : A_0 \times A_1 \rightarrow R$
- We refer to a tuple of actions $a = (a0, a1) \in A = A_0 \times A_1$ as an outcome.
- . The utility function of party expresses the player's preferences over outcomes.
- A strategy $\sigma_i$ for $P_i$ is a distribution on actions in $A_i$.
- Strategy vector, utility on strategy vector.

# Definition for Nash Equilibria(normal form, complete inf. Game)

**Definition 2.3 (Nash equilibria for normal form, complete information games)** *Let* $\Gamma = (\{A_0, A_1\}, \{u_0, u_1\})$ *be as above, and let* $\sigma = \sigma_0, \sigma_1$ *be a pair of strategies as above. Then* $\sigma$ *is in a* Nash equilibrium *if for all* $i$ *and any strategy* $\sigma_i'$ *it holds that* $u_i(\sigma_0'', \sigma_1'') \leq u_i(\sigma)$*, where* $\sigma_i'' = \sigma_i'$ *and* $\sigma_{1-i}'' = \sigma_{1-i}$.

- Naturally extended to the case of extensive form games. (Parties take turn while taking actions).
- Strategy is probabilistic function of a sequence of actions taken so far by players.
- Execution of game is represented naturally via history.
- Utility is applied to the history.

- Another natural extension is to games with incomplete information. (Player has additional information called **type,** only known to itself).
- Additional input. Utility also depends on types (of both players), along with history.
- However, a party cannot necessarily compute its own utility.
- It is assumed that an a priori distribution on the inputs (types) is fixed.
- The expected utility is computed wrt this distribution.

**Definition 2.4 (Nash equilibria for extensive form, incomplete information games)** *Let* $\Gamma = (\{A_0, A_1\}, \{u_0, u_1\})$ *be as above, and let* $D$ *be a distribution over* $(\{0,1\}^*)^2$. *Also, let* $\sigma = \sigma_0, \sigma_1$ *be a pair of extensive-form strategies as described above. Then* $\sigma$ *is in a* Nash equilibrium for $D$ *if for all* $i$ *and any strategy* $\sigma_i'$ *it holds that* $u_i(x_0, x_1, \sigma_0''(x_0), \sigma_1''(x_1)) \leq u_i(x_0, x_1, \sigma_0(x_0), \sigma_1(x_1))$, *where* $(x_0, x_1)$ *is taken from distribution* $D$, $\sigma_i(x)$ *denotes the strategy of* $P_i$ *with type* $x$, $\sigma_i'' = \sigma_i'$ *and* $\sigma_{1-i}'' = \sigma_{1-i}$.

# Extension for the cryptographic model

- Further extension to computationally bounded players.
- Step 1: Model a strategy as interactive probabilistic Turing machine that algorithmically generates the next move given the type and a sequence of moves so far.
- Next, we move to an asymptotic treatment to capture computationally bound behaviour. (We consider infinite sequence of games. Security parameter is given as an additional input to each utility function, set of possible actions and distribution over types.)
- The only strategies which we consider are polynomial in n.
- Relaxed notion of “greater or equal to” to “not significantly less than.”

# Extension to the case of computationally bounded players

**Definition 2.5 (Computational Nash equilibria for extensive form, incomplete inf. games)** *Let* $\Gamma = (\{A_0, A_1\}, \{u_0, u_1\})$ *be as above, and let* $D = \{D_n\}_{n \in \mathbf{N}}$ *be a family of distributions over* $(\{0,1\}^*)^2$. *Let* $\sigma = \sigma_0, \sigma_1$ *be a pair of* PPT *extensive-form strategies as described above. Then* $\sigma$ *is in a* Nash equilibrium for $D$ *if for all sufficiently large* $n$*'s, all* $i$ *and any* PPT *strategy* $\sigma_i'$ *it holds that* $u_i(n, x_0, x_1, \sigma_0''(n, x_0), \sigma_1''(n, x_1)) \leq u_i(n, x_0, x_1, \sigma_0(n, x_0), \sigma_1(n, x_1)) + \mu(n)$, *where* $(x_0, x_1)$ *is taken from distribution* $D_n$, $\sigma_i(x, n)$ *denotes the strategy of* $P_i$ *with type* $x$, $\sigma_i'' = \sigma_i'$ *and* $\sigma_{1-i}'' = \sigma_{1-i}$, *and* $\mu$ *is a negligible function.*

# Our Setting: (informal understanding)

- Two-party protocol
- At each step, the relevant party can make a binary decision: Either abort or continue running the protocol $\pi$ scrupulously. (binary choice)
- We let each player have local history (consists only of the type of the player and its internal randomness).
- The notion of an “action” is extended to include potentially complex algorithmic operations. (algorithm, configuration)
- The outcome of this action is appended to the history of the execution, and the new configuration of the algorithm is added to the local history of the player.

# Formally defined,

**Definition 2.6** *Let $\pi = (P_0, P_1)$ be a two-party protocol (i.e., a pair of Interactive Turing Machines). Then, the* local history *of $P_i$ (for $i \in \{0,1\}$), during an execution of $\pi$ on input $(x_0, x_1)$ and* internal *random tape $r^i$, is denoted by $\mathbf{History}_{\pi,i}(x_0, x_1, n)$ and equals $(x_i, r^i, m_1^i, \ldots, m_t^i)$, where $m_j^i$ represents its $j$th message. The* history *of $\pi$ during this execution is captured by $(m_1^0, m_1^1), ..., (m_t^0, m_t^1)$ and is denoted by $\mathbf{History}_\pi$. The configuration of $\pi$ at some point during the interaction consists of the local configurations of $P_0, P_1$.*

# Fail-stop games

- Party makes strategic decision whether to continue or abort.
- If continue:

  Outgoing message is generated

  New configuration added to local history
- If abort:

  Abort symbol is added to configuration of both parties.

  Protocols run to completion producing local outputs.

# Basic idea

- Utility may depend on all the histories. Convenient to define utility that consider local output of a player. [$\text{Output}_{\pi,i}$ ]
- We investigate the basic Game Theoretic property of protocols whether the pair of strategies ($\sigma_{\text{continue}}$, $\sigma_{\text{continue}}$) is in a (computational) Nash equilibrium in fail-stop games, with respect to a given set of utilities and input distributions.

**Definition 2.7.** (*Nash protocols*) Let $\mathcal{D}$ be a set of distribution ensembles over pairs of strings, and let $\mathcal{U}$ be a set of extensive form binary utility functions. A two-party protocol $\pi$ is called Nash Protocol with respect to $\mathcal{U}$, $\mathcal{D}$ if, for any $u \in \mathcal{U}$ and $D \in \mathcal{D}$, the pair of strategies $\sigma = (\sigma^{\texttt{continue}}, \sigma^{\texttt{continue}})$ is in a computational Nash equilibrium for the fail-stop game $\Gamma_{\pi,u}$ and distribution ensemble $D$.

# On subgame perfect equilibria and related concepts.

- Solution concept for extensive for games. [not encumbered by “empty threats.”]
- We note that in our limited case of fail-stop games any Nash equilibrium is subgame perfect.
- Indeed, once one of the parties aborts the computation, there is no chance for the other party to “retaliate”; hence, empty threats are meaningless.
- Some variants of this notion that are better suited to our computational setting have been recently proposed.

# Privacy and Correctness in Game Theoretic View

# Privacy (Basic setting)

- A protocol is private if no (failstop) PPT adversary is able to distinguish any two executions where the adversary's inputs and outputs are the same, even when the honest party uses different inputs in the two executions.
- Goal is to define a set of utility functions that preserve this property for Nash protocols.
- Restrict to input distributions over triples of inputs. Input given to one of the parties is fixed, input of the other party is uniformly chosen from the remaining pair. [captures the strength of cryptographic (semantic) security; other party input is one of two possible values, cannot tell which one.]

# Privacy (Defining utility)

- One could define privacy by having each party gain whenever it learns something meaningful on the other party's private input.
- Seems that it is better to make a party lose if the other party learns anything about its secret information.
- Intuition: It must be worthwhile for the party who holds the data to maintain it a secret. Having the other party gain any profit when breaking secrecy is irrelevant, since it does not introduce any incentive for the former party to prevent this leakage.

# More formally

- For each n we consider a sequence of input distributions for generating inputs of this length. Each distribution is denoted by D and indexed by triples ($a_0$, $a_1$, b), and is defined by picking $x \leftarrow (a_0, a_1)$ and returning (x, b).

**Definition 3.1.** (*distribution ensembles for privacy*) The distribution ensemble for privacy for $P_0$ for a two-party function $f$ is the ensemble $\mathcal{D}_f^{\mathsf{p}} = \{D_{f,n}^{\mathsf{p}}\}_{n \in \mathbf{N}}$ where $D_{f,n}^{\mathsf{p}} = \{D_{a_0,a_1,b}\}_{a_0,a_1,b \in \{0,1\}^n, f(a_0,b)=f(a_1,b)}$, and $D_{a_0,a_1,b}$ outputs $(x, b)$, where $x \stackrel{R}{\leftarrow} (a_0, a_1)$.

- For every $(n, a_0, a_1, b)$ and for every PPT algorithm B, let the augmented protocol for privacy for $\pi$, with guess algorithm B, be the protocol that first runs $\pi$ and then runs B on the local state of $\pi$ and two additional auxiliary values.
- We assume that B outputs a binary value, denoted by $guess_{\pi}^{p}{}_{Aug,B,}$ where $P_i$ is the identity of the attacker.
- This value is interpreted as a guess for which one of the two auxiliary values is the input value of the other party.

# Defined utility

**Definition 3.2.** (*utility function for privacy*) Let $\pi$ be a two-party protocol and $f$ be a two-party function. Then, for every $a_0, a_1, b$ such that $f(a_0, b) = f(a_1, b)$, and for every guessing algorithm $\mathcal{B}$, the utility function for privacy for party $P_0$, on input $x \in \{a_0, a_1\}$, is defined by:

$$u_0^{\mathsf{p}}(\mathbf{History}_{\pi^{\mathsf{p}}_{\mathsf{Aug},\mathcal{B}},1}(x, b, n), a_0, a_1, b) \mapsto \begin{cases} -1 \text{ if } \mathsf{guess}_{\pi^{\mathsf{p}}_{\mathsf{Aug},\mathcal{B}},1} = g \text{ and } x = a_g \\ 0 \quad \text{otherwise} \end{cases}$$

# Understanding

- If the history of the execution is empty and inputs of the parties are taken from a distribution ensemble for privacy, then $u_0^p$ equals at least −1/2. [$P_1$ can only guess x with probability at most 1/2.
- Therefore, intuitively, it will be rational for $P_0$ to participate in the protocol (rather than to abort at the beginning) if and only if the other party cannot guess the input of $P_0$ with probability significantly greater than 1/2.

**Definition 3.3.** (*game theoretic private protocols*) Let $f$ and $\pi$ be as above. Then, we say that $\pi$ is *Game Theoretic private for party* $P_0$ if $\pi^{\mathsf{p}}_{\mathsf{Aug},\mathcal{B}}$ is a Nash protocol with respect to $u_0^{\mathsf{p}}$, $u_1^{\mathsf{p}}$ and $\mathcal{D}_f^{\mathsf{p}}$ and all valid PPT $\mathcal{B}$.

- A protocol is Game Theoretic private if it is Game Theoretic private for both parties.

**Theorem 3.4.** *Let $f$ be a deterministic two-party function, and let $\pi$ be a two-party protocol that computes $f$ correctly (cf. Definition 2.2). Then, $\pi$ is Game Theoretic private if and only if $\pi$ privately computes $f$ in the presence of fail-stop adversaries.*

# Proof of theorem

# Correctness in Game Theoretic View

# Basic setting

- We formulate utilities.
- We show that a protocol correctly computes a deterministic function if and only if the strategy that never aborts the protocol is in a computational Nash equilibrium with respect to the set of utilities specified.
- The parties get high payoff only if they output the correct function value on the given inputs (types), or abort before the protocol starts; in addition, the players get no payoff for incorrect output.

**Definition 3.5.** (*distribution ensemble for correctness*) Let $f$ be a deterministic two-party function. Then, the distribution ensemble for correctness is the ensemble $\mathcal{D}_f^{\mathsf{C}} = \{D_n^{\mathsf{C}}\}_{n\in\mathbf{N}}$ where $D_n^{\mathsf{C}} = \{D_{a,b}\}_{a,b\in\{0,1\}^n}$, and $D_{a,b}$ outputs $(a, b)$ w.p. 1.

- A fail-stop adversary cannot affect the correctness of the protocol [plays honestly with the exception that it may abort.] An incorrect protocol in the presence of fail-stop adversary implies that the protocol is incorrect regardless of the parties' actions.
- Upon receiving an abort message we have the following:

(i) The honest party already learnt its output. So, correctness should be guaranteed. Or,

(ii) The honest party did not learn the output yet, for which it outputs $\perp$ together with its guess for the output.

Note: The guess is different from the case of privacy as here, we assume that the protocol instructs the honest party how to behave in case of an abort.

# Modelling utility

- The parties gain a higher utility if they output the correct output, and lose if they output an incorrect output. Therefore, the continue strategy would not induce a Nash equilibrium in case of an incorrect protocol, as the parties gain a higher utility by not participating in the execution.

# Defined utility

**Definition 3.6.** (*utility function for correctness*) Let $\pi$ be a two-party fail-stop game as above. Then, for every $a, b$ as above the utility function for correctness for party $P_0$, denoted $u_0^{\mathsf{C}}$, is defined by:

- $u_0^{\mathsf{C}}(\mathbf{History}_{\pi,0}^{\phi}) = 1.$
- $u_0^{\mathsf{C}}(\mathbf{Output}_{\pi,0}, a, b) \mapsto \begin{cases} 1 \text{ if } \mathbf{Output}_{\pi,0} = f(a,b) \\ 0 \text{ otherwise} \end{cases}$

where $\mathbf{History}_{\pi,0}^{\phi}$ denotes the case that the local history of $P_0$ is empty (namely, $P_0$ does not participate in the protocol).

**Theorem 3.7.** *Let $f$ be a deterministic two-party function, and let $\pi$ a two-party protocol. Then, $\pi$ is a Nash protocol with respect to $u_0^{\mathsf{C}}$, $u_1^{\mathsf{C}}$ and $\mathcal{D}_f^{\mathsf{C}}$ if and only if $\pi$ correctly computes $f$ in the presence of fail-stop adversaries.*

# Proof of theorem