[SQUEAKING]

[RUSTLING]

[CLICKING]

ROBERT M. TOWNSEND: So let me just say that, shockingly, we have arrived at lecture 8, which means we're well more than halfway done with the lectures. So today is continuing the theme of trying to bring computer science and economics onto the same page. And the juxtaposition is everywhere today, so "Mechanism Design and Incentives" has to do with the way economists think about trust versus "Protocols and Notions of Trust" in computer science.

The longer title is "Private Information, Incentives to Report and Take Actions"-- as in mechanism design versus illustrative Byzantine Generals problem that utilizes a different notion of trust to emphasize the difference between economics and computer science in the way we think about algorithms, which is not to say there isn't a middle ground. I'm trying to avoid a bipolar view, but that is the way the lectures are laid out.

So the outlined is information constrained allocations. And I'll do it through an example, Although the principles generalize. And we'll think about implementing this without a planner in the jargon of economics, rather, using the tools of computer science. So that helps mitigate the bipolar view of economics versus computer science. We will utilize the tools of computer science very heavily within the mechanism design problem.

And then we'll go on to talk about algorithms for validation and so on and readdress this difference by an illustrative paper of Steve Morris and Hyun Shin on the Byzantine generals problem.

So first half-- Information Constrained Allocations vis a vis this insurance example-- how to implement without a planner using the tools of computer science. So to be specific, I will talk about an agrarian economy. But that said, we can think about this notation for the same model as applying much more generally to today's financial markets.

So it's a pure exchange economy. There's only one period for now. There's two agents, labeled 1 and 2, and a K-dimensional vector of endowments. The endowment of agent 1, the villa, so to speak, is seen by that agent alone. So shocks are private to the agent. And we can parameterize this endowment, which is e1 for agent 1, and e is endowment as a function of epsilon by a simpler notation, where theta is this parameter theta, taking realizations in some larger set with probabilities p of theta. So that's for the villa.

And agent 2 is like a central monastery. Agent 2's endowment for simplicity is public. For that matter, agent 2 is going to be risk-neutral. Agent 1 is going to be risk-averse. And again, we have this K-dimensional vector, publicly observed vector of, quote, "endowments" for the monastery.

So these two agents are going to agree to some rule for allocating resources-- some rule f, function f, mapping a message sent from agent one to agent 2 into positive and negative transfers of each of the items in this K-dimensional vector. It will be in the notation as if the villa is paying a tax, as if the transfer, if positive, is going from the via to the monastery. But transfers can be negative. This is, after all, an insurance example. The agent has a random endowment, is risk-averse, and the monastery agent 2 is willing to undertake that insurance. But the problem will be how to mitigate the damage caused from the private information if you can.

So they agree to a resource allocation scheme, which works as follows. Villa 1, agent 1 waits to see the output vector parameter theta, then sends a message m to the monastery, and the allocation would be f of m. Let's focus on the decision problem of the agent about what message to send. So the agent knows theta by now prior to sending the message and chooses some message m so that the transfer from the agent to the monastery would be f of m. So that just subtracts off of output.

So what message to send? Well, let's suppose there is a unique maximizing message, and let's put a star on it. So m star of theta is the actual optimized message the agent would send given the set of messages which are possible in the allocation rule f. So this statement, 83, is a simple statement of maximization that the chosen message is at least as great in utility as choosing any other arbitrary message. In fact, if it were unique, then this is a strict inequality for any message other than m star-- so just a statement of the maximum.

And in particular, if we took m on the right-hand side to be a particular message, any message will do. 83 still holds. But let's consider m to be what the agent would have sent in the counterfactual situation where the theta were theta tilde rather other than what it really is, namely theta. So we just substitute this chosen alternative message m equal m star of theta into equation 83. And we end up with this equality. So that inequality is a simple substitution.

So now we just do some-- this is a very broad class of games. I haven't told you what the set of messages looks like. I haven't told you what f's, how they're chosen from some domain. It's a very broad mechanism design problem. But we can come up with a simpler scheme. And now I'm about to describe this alternative scheme.

And in it, the message space is now restricted to the set of possible values of theta. That doesn't mean that we're going to require the agent tell the truth. He just outputs, high. It could have been low or vice versa. Lying is perfectly possible. But he can only announce possible values of theta. And it's common knowledge what the set of realizations of theta is. So that's enforceable.

And we do something with the f. Namely, we invent a new allocation rule g. and g logically just maps these announced messages about theta into an allocation. So here, we're doing it for this maximizing message the agent would have sent if his parameter were theta tilde, which is really a composite function over two things. It's over f, the allocation rule, and over the optimized maximized strategy the agent would employ in the original game. And we collapse that down to g.

So now the agent says, I know what I would have done if I had a certain value theta tilde. So I'll just let the computer do that for me. I'll just load in my maximizing strategy and just say, here's my theta. Then the m star of theta behind the scenes would be generated, and it would be mapped under f to g. We'll just now adopt g as, in the end, the allocation rule, this alternative allocation rule in this new mechanism.

Well, again, if you start doing this substitution, this is now g of theta. This is g of theta tilde. So we have this new equation 85. And so 85 just simply says, if the parameter value were theta and you said so, that would dominate weekly in utility terms having theta and announcing something else, theta tilde.

So this looks like a truth-telling constraint. But that's really very treacherous way to put it because we're not requiring truth telling. We're inducing truth telling without loss of generality as a way to represent the outcome in a game where there's private information. And, of course, if telling the truth is what the agent does, then the allocation will be g of theta, which is exactly what would have happened before under the old allocation rule f with the optimizing behavior m star.

So we're going to achieve exactly the same outcomes in this general game with f and capital M as we achieve now. The thing is we know how to search for optimal mechanisms now because all we need to do to capture the private information is impose, without loss of generality, this constraint 85.

So to find the optimized allocation rule, we do the usual thing. We're going to maximize a lambda-weighted sum of the ex ante expected utilities of the agents subject not just to the resource constraint, but to this incentive constraint. So this is lambda 1 for agent 1. It's ex-ante, so we're taking expectations over theta with the associated probabilities. Again, by our normalization, what agent 1 gives up, agent 2 gets. Well, again, it's easy to lose sight of it. These could be vectors. And we've already done 87.

So there are various possible ways to implement this. You can think of the agent as sending a message. You can think of them agreeing ex-ante to a list of possible transfers that solve this problem for each and every theta. And then the guy just, here it is. Here's what I'm transferring today because it's 1 to 1 from the theta to the transfers.

Now, this is where I go back and forth. If there were only one good, this constraint could be pretty damaging because there's only one period and one good. More is preferred to less. So you can see this would say, well, g of theta is less than or equal to g of theta tilde. But that's true for any theta-theta tilde combination. So effectively, g must be a constant. It cannot depend on theta. And if you're trying to make both better off, you'll be reduced to autarky. Now, with more than one good, there's a trade-off among goods. So this doesn't necessarily reduce to autarky.

Another way to potentially enhance the set of possible allocations is to start considering randomness. And if there's differential risk aversion, depending on your theta, there's a trade-off between high means and high variance. And you can keep the agent away from saying something under which he would get a transfer on average because it's also associated with a lot of risk.

So in general, we'll revert to these lotteries. Instead of saying, you say theta, and you get a transfer, it's going to be you say theta, and you get a transfer with a certain probability pi, and tau is the transfer. We take this program 6 and convert it to program 7.

Same thing, essentially. Maximize weighted sums of utilities. But now, for theta, we have this lottery index by theta, pi tau of theta. And we're summing over, say, a finite, discrete large number of taus on both sides. This is give up. This is get. This is the new version of the so-called truth-telling constraint. Now, I haven't rederived it. But intuitively, it's not that hard to end up with this as the logical consequence of private information when the mechanisms are randomizing over outcomes.

In my head, when I'm proofreading or writing, I always go through this exercise of, what is the actual parameter value? And what is the counterfactual? So in this case, theta is the real value. You say theta. Theta remains the true value. But you could say theta tilde.

STUDENT: I missed the step. Why did we move to a random mechanism from before?

ROBERT M. TOWNSEND: Well, there's a couple of reasons. In general, it can allow more trade than deterministic allocations. When there's only one good, there's no trade-off. More is preferred to less. With the lottery, you have a mean variance trade-off, and you can exploit that sometimes.

Another reason is this is quite computable. This is a linear program because the objects are these probability numbers. And they're entering the objective function linearly. And they're entering the constraints linearly.

I mean, we're so used to thinking of choosing tau within the utility function. Now, with the lotteries, we're choosing the weights on that outcome. So that's kind of a trick. But it does mean when we're stalled out and we can't characterize the solution as well as we might like to do analytically, we can resort to linear code to compute these information-constrained optimal allocations. And programs like Gurobi, which is open software, can handle hundreds of thousands of variables and thousands of constraints and solve it relatively quickly on your laptop.

STUDENT: And the W should be a theta?

ROBERT M. TOWNSEND: The W here is this constant endowment vector of--

STUDENT: Ah, that's not theta.

ROBERT M. TOWNSEND: No.

STUDENT: So that's [INAUDIBLE].

ROBERT M. TOWNSEND: Yeah. This is agent 1 with theta. This is agent 2 with W.

STUDENT: I see.

ROBERT M. TOWNSEND: And again, I'm not going to do that today. But you could imagine two active agents, each with his or her own random outcomes. So it could still be an agrarian economy where farmers are growing rice, and they take actions, and they have randomly determined harvests as a function of temperature and rainfall. But it equally applies to financial markets where they have assets.

And the assets have randomly determined yields. And somehow, they get to see something that external agents don't see. But we could have them announce these unobserved states. So it applies to high-valued financial markets-- portfolios of the agents being private and subject to these unobserved random shocks. OK.

Now, again, I'm not going to do much about this. So if there were one good, it's easier to think about it that way than it's a scalar economy, and we're talking about insurance. So the logical target is like a full insurance allocation, where the agent with the low outcome would like an indemnity and is willing, in turn, to pay a premium if his or her output were high.

So what's the incentive to lie? If you're high, you would be paying a premium. So you might prefer to get the indemnity. So the binding incentive constraint on such a problem would only apply to the high-theta agents who would prefer not to pay the premium. And the guy receiving an indemnity doesn't have any binding constraint.

I mention that because, in general, you could imagine enhancing the mechanism to allow pre-transfer displays. I mean, if you're claiming something about your output after all, then, in principle, you could show it, like, show me your cards. And if your endowment is low, you can't show stuff you don't have. In this case, it wouldn't be a worry. But in general, you could enhance these mechanisms to allow displays.

So far, it's just been one period with this allocation scheme-- not so much a fixed rental for the land as a schedule of what rent to pay, but only one period. So if there are multiple periods, we can extend it pretty easily. We'll talk about day date t, taking on values 1 or 2, theta sub t being the privately observed endowment of the villa agent 1. At date t, W sub t now being endowment of agent 2 allowed to vary with time, but not random otherwise.

And we can have these Markov probabilities on the theta. What's the probability of the first one? Theta 1. What's the probability of the second one conditioned on the first one? And write down a little bit more notation, although it's the same idea as before. We're going to maximize a weighted sum of utilities of the agents. There's just more terms in the utility outcome function for each agent to capture the first period with the lottery, and also the second period with the lottery. It's a lot of notation.

So in the second period, this is the probability of tau given the realization of theta at date 1 and the realization of theta at date 2. This is actually date two. So theta 2 is in agent 1's utility function by the normalization, quote, "paying out tau." And then these are the overall probability of theta 2 given theta 1 times the probability of theta 1 is the joint probability of theta 1 and theta 2. We discount by beta, although it could be 1. No harm done for agent 1. And this is this longer expression for agent 2.

Now, if there's anything interesting in this slightly enhanced model, it's this guy here that we've now put a theta 1 into the allocation rule at date 2. And that's going to be very helpful to allow some gains to trade.

But before I get into that, really, the proof of it has to do with these incentive constraints. So it's like a dynamic program. The world ends at date 2. So let's figure out what's happening. Somehow, theta tilde 1 was announced. Just take it as a given, whatever it was, truth or a lie. It's indexing the allocation rule.

So now the agent-- remember my routine here. Theta 2 is the actual. You could say the actual. It's still the actual, but you could lie about it and say, theta tilde 2. So this ensures you'll weakly tell the truth when the parameter value-- and this is true for all possible histories theta tilde 1 and all possible theta 2's. It says it in the text, but it would be better to write that out explicitly just to remember. It's not just one constraint. It's a whole family of constraints.

And what happens at date 1? Well, with this, you're sure the agent will, quote, "tell the truth in the second period." So now let's take that as a given. The incentive constraint for date 1 takes as given that when you have theta 2, you will say theta 2. That's inherited from 96. What this is running over is the possible lie about the parameter at date 1. Theta 1 is the actual value. And you should say so rather than theta 1 being the actual value and the villa saying theta tilde 1. So this is the pair of these constraints' families is going to ensure you'll tell the truth in both dates.

So what happens if there are no incentive constraints? Then we're looking at the usual risk-sharing problem. Agent 1, risk-averse; agent 2, risk-neutral. We know which way the insurance should flow. Not only that, we know from the risk-sharing problem that it decomposes into essentially a static problem. You just take the level of community resources, pool them together, and implement the optimized risk-sharing rule. The amount there can vary, but the rule does not vary. And so even though with all these dates, and states, and everything else, we, without additional constraints, got essentially static risk-sharing rule.

Will that work here? No, because of the private information. Again, if there's only one good, the agent would always claim theta to be low to get the indemnity. And it's not rocket science. Agent 2, the monastery, knows that. So they could not possibly be trading anything. So full risk sharing is not feasible.

With the two periods, though, something more than autarky is possible. So even though you can't do much at the second date if there's only one good, effectively it's a constant. What is that constant? Well, it could be a bit like borrowing and lending. When you think about it, the option is on the table to either borrow or lend at a known interest rate. And the person on the other side doesn't know the income, but lets the agent choose whether to borrow money or deposit money, say, in the bank with a principal.

So we think about the standard borrowing-and-lending thing. If your income were low, you're short of resources, you have a high marginal utility for consumption. So you would prefer to borrow granted the commitment to pay back in the next date. And we're assuming there's full commitment.

You could say your endowment is high, and you want to be a lender. But that just goes the wrong way. You're subtracting resources away when you're short and getting them when you're well endowed in the next period. So although we typically don't think about it that way, borrowing and lending is incentive-compatible, even with private information.

But is it optimal? No. Borrowing and lending is not the optimized solution to this problem. Well, why is that?

Well, go through what I was just describing. Suppose you strictly prefer to borrow when your income is low, you strictly prefer to lend when your income is high. So that means the so-called incentive constraints apply with a strict inequality in the proposed solution.

So you have the full solution with the incentive constraints. We came up with a feasible allocation, which is incentive-compatible. But it has both of the truth-telling constraints not binding. You have a strict preference over what you're doing.

And if they're not binding, suppose it were optimal. That means if that's the optimized solution without incentive constraints, it should be full insurance. But full insurance is not feasible. So it's a little bit of a proof by contradiction that the optimized solution cannot be the borrowing-and-lending solution. What will it be? It's a hybrid scheme that embeds the risk-sharing aspect with the intertemporal trade-off.

The intertemporal trade-off is, what happens at the second date depends on what happened at the first one, or, at least, what was said about it. That's the tie-in. And that's the reason we're having this theta tilde 1 entering into the--

STUDENT: Sorry. I didn't understand why. Why are the incentive constraints strict when we do the borrowing and lending? Why do they have to be strict in that case?

ROBERT M. TOWNSEND: It's more, what if they were strict?

STUDENT: So I agree that if they were strict, it's not optimal. But why was it true that whenever we implement a borrowing and lending, they have to have--

ROBERT M. TOWNSEND: Strictly concave utility and the endowment being a scalar, being low or high, so the marginal utility of consumption is high when your income is low. So you'd like to get something incoming.

Well, again, what is the meaning of it? It means if we're trying to implement, at least in this environment, an allocation mechanism that respects private information, we would expect it to not be pure credit. We would expect it not to be pure insurance either. It would be a financial contract, which is like a hybrid blend of credit and insurance. And we see those kinds of things in practice. We see flexible loans, for example, where it may be possible to defer repayment, depending on what is announced by the client, for example.

On the other hand, it would probably be a bad idea to separate the insurance company and say, only insurance companies can offer that stuff from the borrowing and lending and say that only banks can do it because it's one agent with an underlying decision problem. And those two pieces really need to be coupled together in a pretty careful way. Otherwise, one or the other of those parties could-- the insurance company always able to get a payoff to pay off the loan, then the lending company is going to go bankrupt. OK.

Implementation without the planner-- well, the planner is a reference to the maximization problem. In economics, we refer to these things as planning problems. But it does conjure up a third party implementing the thing. We really don't need a third party. We don't need a trusted planner.

The information-constrained allocation rule is a code. You can call it, say, a smart contract. I'll come back to that in a minute. And the agents enter into it voluntarily, having understood how the code works. They hash it up. We'll come back to that next time we talk about encryption.

And so there's a public display of the commitment to carry out the plan. This is what we agree to. Hopefully, some judge would be persuaded that it was all voluntary.

After the fact, you could imagine one or the other of these guys complaining, like, oh, no, I didn't really mean it. I don't want to have to pay, like the trade fails we talked about. But again, it would be clear that they had agreed to it.

In terms of where the resource is coming from, this depends a bit on the context, but you can imagine calculating from the smart contract the maximum amount that would be required to be paid out. Not that it necessarily will be, that it could be run over positive, negative, what's the worst case scenario for me? That goes into escrow. So now that's tied up. You don't have any control over it. If you agree to it, before this thing even runs, they put stuff in escrow in a way that guarantees performance. So that handles the limited-commitment problem.

And again, if you're a little leery about making all these messages and having them recorded on the ledger and so on and so forth, then we can resort to encryption. And I will talk about that next time. So these messages can be kept private. The key part of the encryption is that the code can operate on encrypted messages. It doesn't need to see the actual numbers. It will do the right thing.

And where are all these two periods or more? There's a history here of messages, which is key to the optimized allocation rule. So those messages are also recorded. They're part of the database. We talked about the history of transactions on Bitcoin as squirreled away in these Merkle tree archives of data. In this case, you would also want to be storing the history of messages securely with the hashes and so on.

And finally, a bit about layer 1 and layer 2-- when we think about Ethereum, it reads as if anytime the code as a node is going to proceed with a transaction, that it has to be validated by the community. Well, here, what I want to do is distinguish being validated and run on all the EVMs of all the nodes. Yes, they've all agreed to the code, and it runs on everyone's computer.

But there is no need to be executing the mechanism step by step, line by line, over time,et cetera, and throwing that out for validation on the premise that they know what they've agreed to, and they understand how it operates, and so on. So you end up with diagrams like this, where the smart contract is here. And these agents-- in this case, two of them sending messages from the permitted message spaces about underlying unobserved objects that enter into the smart contract. The smart contract is, for example, the one we derived just now in that economy. And the messages lead to obligations to transfer or receive money.

So we need the ledgers. But this one got deleted. Here's the ledger. There's the balance sheet up there. So they all have their balance sheets on Ethereum. And at the end, when the mechanism is executed, they're making transfers across the balances consistent with the smart contract.

This is called layer 2 because-- again, I'm repeating myself-- it's not all being done in real time on Ethereum with ether being used up for the validation. This is an amusing typo, actually worse because it was deliberate on my part. This should say off-chain in the pink. And below it in the black and white should be on-chain-- the chain referring to the ledgers and the blockchain. Whereas the contract is external, being run effectively without the validation.

And I just assume that the center of the Earth was the smart contract. So it should be on-chain, but that's not the right nomenclature for this problem.

OK. All right. So let's come to protocols and notions of trust. First, a look at the validation algorithms in Ethereum, and Bitcoin, and others, and then, finally, a tension between following a protocol as in computer science versus incentives in the mechanism design problem. So for alternative protocols, we have mentioned Bitcoin, and we will come back to Bitcoin again next time. Bitcoin uses up electricity. And it's not fast either.

Why? Essentially, the idea is that whoever solves the cryptographic puzzle the first is the one to, quote, "validate" the batch of transactions that that's up for validation. But it's random who's going to solve it first because you can only solve it with trial and error. So most people are honest. So what are the odds that a randomly chosen validator will be one of the nefarious evil nodes? So that's the intuitive logic of the Bitcoin validation, but it is using computers, as you all know, to solve these puzzles.

There are other algorithms that are faster and less costly. So we have Proof of Work is the way people refer to Bitcoin. We have Byzantine Fault-Tolerant, BFT, algorithms that can also reach consensus, even if there are adversarial nodes or if nodes drop from the network.

And note, nefarious and faulty computers are being used synonymously here. The spirit of this is parallel computing, where you have multiple computers running, but some of them could malfunction. So how many computers do you need? If you're willing to say what fraction of them can fail, then you need, in this case, for the practical Byzantine Fault-Tolerant algorithms, you need the number of failing nodes times 3 plus 1 replicates to actually know what the truth of the outcome is by comparing the array of answers.

This algorithm also chooses a leader in a round-robin fashion. So you have of chosen one who takes the lead, but it'll be a different person next time around. And those guys are chosen from a membership list. So this could be more of a closed rather than open, public validation. You could use it within a company, for example.

An extreme version-- I don't think it's in the slides-- I find amusing. It's called proof of authority. So there's one guy approving everything. Could be a central bank. And so whether or not validation is really time-consuming and a barrier really depends on the particular validation algorithm that you're using.

Proof of Stake is another one where the validators are chosen at random. And then that's followed by voting. But if you hold more coins, you have a bigger stake in the outcome, so you get a bigger vote.

And finally, another one is the one developed by Ripple and Stellar, Federated Byzantine Agreement. They used to be the same company. They kind of split up. One is not for profit. The other is profit-oriented.

Mazieres worked for Stellar at Stanford. He created this Federated Byzantine Agreement. So this is interesting in terms of trust. Each node decides lists the other nodes that it trusts, not necessarily all of them-- the neighbors, business associates, other companies, et cetera.

And that's called, for that node, a quorum slice. And the overall quorum, which means you can go ahead, it's official has to do with the way these slices overlap. So if there's not much trust in the system, then this algorithm actually doesn't execute very much. But it can be very powerful. So yeah, the whole goal here is to be able to execute stuff quickly for large-scale problems where there is a lack of trust in the outcome.

Now, I must say, editorialize a bit-- and this is a preview of coming attractions-- the very first thing that economists got interested in with respect to Bitcoin was whether Satoshi's algorithm was really going to constitute a Nash equilibrium. And what I mean by that is the number of miners has often been very few, very, very concentrated. Not only that, they tend to do-- they handle the risk. Only one wins right and gets more Bitcoin. So they've entered into syndicates that pool the risk, so they can share the rewards. But like Adam Smith said, when two or more gather together, beware of the outcome.

So I'm just picking up a bit on this tension. Economists naturally got involved in the incentives associated with the validation algorithm, not necessarily to just critique it, but potentially to make it more robust. I mean, that said, to my knowledge, the Bitcoin algorithm has never failed, even though-- well, let me take that back. None of this is on the slides.

There is this notion that you can prioritize validation. Pay value to a validator to prioritize the block that you're submitting. And again, you could imagine that might not be the best idea. It's like bribing a validator. But so on it goes.

So let's just think about a little more formally the incentives to follow an algorithm, combining the incentives of mechanism design and game theory in the context of implementing an algorithm. And we're going to do this in the context of the Byzantine Generals Problem.

So this is overview slide. This is a classic problem. I don't know why it's called Byzantine, but that's the reference to it-- coordinating a successful attack on an enemy. The enemy may or may not be prepared. A coordinated attack will be successful if it is coordinated and both generals are attacking, and if the enemy is unprepared. But the attack fails if only one general is doing the attacking. And it requires, for success, both generals to attack. One is better informed than the other, so the first general can send messages to the second about what he knows about the outcome, the likelihood that the enemy is prepared.

But here's the catch. These messages are noisy. They don't arrive with probability 1. Rubenstein has an equally famous article called "The Email Problem," which is closely related.

So Morris and Shin-- and I'll show you this in the slides-- show that the strategic maximizing behavior in an explicit information game without commitment is inconsistent with the naturally prescribed protocol. And I'll go through that protocol.

But there is an alternative protocol with commitment-- in some sense, this slide is premature because I haven't laid out all the notation-- prevents the second general from trying to confirm an incoming message. So the message from the first to the second did arrive. The enemy is unprepared. The attack can be successful. But to validate the message, the second general is supposed to send a return message back. So the first general knows the second one got it, and they can coordinate on the attack. It's very intuitive.

The counterintuitive-- the twin problem here is going to be that the intuitive message-and-action protocol is not consistent with rational maximizing behavior on the part of the generals. And equally counterintuitive, you want to shut down some of the messages in order to get something that works almost all the time and works well, namely this confirmation message.

Imagine you're two divisions of an army, each commanded by a general. They're on these hilltops, looking into the valley. And the enemy is below in the valley. The commanding general of the first division has a highly accurate intelligence report informing him of the state of the readiness of the enemy. It's clear that if the enemy is unprepared and both divisions attack at dawn, they will win the battle. But if the enemy is prepared or only one general attacks, the venture will fail.

First-division general is informed the enemy is unprepared. If that happens, he has a signal indicating it's very likely the enemy is unprepared. He will want to try to coordinate a successful attack. But these generals can only communicate by messengers in this picturesque paragraph here. And unfortunately, the messenger may get lost or, worse yet, be captured by the enemy, meaning the message does not arrive.

OK. So suppose delta is this probability the enemy is prepared. And 1 minus delta is the probability the enemy is unprepared. The first general has this signal. He knows which event we're in, the delta or the 1-minus-delta branch. Now, that doesn't mean that he knows for sure the enemy is unprepared. He just, in one case, thinks it's likely the enemy is unprepared. And the other case, it's likely the enemy is prepared.

And the first general, knowing which contingency it is, the second general does not. And if the first were to send a message to the second one, that could get lost with probability epsilon. So we're going to assume the messages are almost surely going to arrive, but not with probability 1. The epsilon is a small number. It's smaller than delta. And let's assume there's no limit to the number of messages that can go back and forth. Although I think it's Borel-Cantelli lemma tells us, with probability 1, some message is going to get lost at some point.

So here's the naive message protocol. General 1 receives a signal, sends the message if the enemy is unprepared. General 2 may or may not get the message. However, if received, the general 2 replies back with a confirmation message. And so on it goes means that if the confirmation is received, then the first general sends a message back to say, I know that you know that we're ready to go. And that message could get lost, or this could go on for several. Iterations, this back and forth.

So Steve and Hyun have a table indexed by n and m, where n refers to the first general has up to that point sent n messages. The second general has sent m messages.

So if the enemy is prepared with probability delta and the first general knows it, then under what I was just describing, no message is sent. There's nothing to coordinate. We don't want the attack to happen. So neither is sending a message-- 0, 0.

But for the rest of these three lines here, 1 minus delta is the probability the enemy is not prepared. And general 1 sends message, but it could fail to arrive with probability epsilon. Or it does arrive with probability 1 minus epsilon. The confirmation message is sent back from 2 to 1. So in this case, they both sent a message. And that happens with this probability. Hopefully, you're getting the idea here. Now the confirm the confirm message, but that could get lost. The 1 minus epsilon squared refers to all the successfully transmitted and received messages up to a certain point. And then the epsilon refers to a potential point of failure.

So let's suppose that attacking when the enemy is prepared is a disaster, a large negative number, m minus m, and that they both have a payoff of 0 if they're not attacking. And they have a payoff of 1 if the attack is successful. So they are, after all, just brigades in a common army. And the social objective here is to defeat the enemy. So they both lose minus m if the attack fails. For whatever reason, they both lose. I'll come back to that slight modification momentarily.

So the theorem is if the communication system is sufficiently reliable, then the optimal action profile I'm about to tell you has a coordinated attack happening almost always when the enemy is unprepared. So it has very, very good characteristics. There's a lot of ifs in this statement.

And if the communication means epsilon is small, and what is this optimal action profile? So the optimal action profile uses the naive communication protocol of this back-and-forth messaging system, would have the first general attacking-- this is the action-- whenever the enemy is unprepared, even if he has not received a confirmation message from the second general.

And as for the action of the second general, he attacks, even if he received only one message from the first general. He doesn't wait for any reconfirmation of his own message back to the first one to arrive back to him. So in this case, the coordinated attack does occur with overall probability associated with the enemy being unprepared and that this message from the first to the second general arrives.

Remember the action of the second general here. He attacks when he has received the message from the first-division general. Even if the first guy is attacking, the second needs the message from the first one. So just a footnote, as it were, in Morris and Shin's paper is a simple proof. The optimal action protocol is what I read out just now.

And this is a bit tedious, but what the overview of it is considering two nearby strategies. One, the first-division general attacks whenever the enemy is unprepared, and the second general attacks always. And these would be the payoffs because the enemy could be prepared-- in which case, they both suffer minus m. But by chance, the enemy is not prepared and they're both attacking, so that's a payoff. You do a little more algebra on that.

The next line is what we claim, namely the first general attacks whenever the enemy is prepared, sends a message to the second. The second-division general attacks if he has received that one message, and that has this expected payoff. The third neighbor on the other side is each general attacks if he's received at least one message, which means that the first general has to get a confirmation back from the second one. And again, you're just multiplying probabilities by outcomes. So I've reconfirmed this umpteen times. I'll resist the temptation.

I even wrote out more explicitly the algebra underlying the third strategy. So it turns out, when epsilon is small and m is large, the second strategy dominates the near neighbors, and it will dominate every other possible strategy that you could come up with. So it gets us what we want.

But the remaining difficulty is this optimal action protocol is sensitive to strategic concerns. It works as long as the generals follow it.

Now, you see the spirit of this. We already went through these validation program protocols for the blockchains, right? And it's assumed that people will follow those protocols. So here, the analogy is the generals are going to follow this optimal action protocol, maybe. But would they want to? Is it in their self-interest?

It's a cute line here. "Perhaps battle orders instruct them to follow the protocol, and generals always follow battle orders." But anyway, suppose the first-division general knows the enemy is unprepared, sends a message to the second, but does not receive a confirmation. Then he believes, with a certain probability, that his own message never got there. And since it didn't, under the optimal-action profile, the second general is not attacking.

Now, where is this number coming from? So we're going to do a Bayesian thing here. What is the probability that this message never got there? But it could have gotten there. So we do the weighted average-- this probability that it doesn't get there, again, repeated in the denominator and the other event that it did get there, and the second general sent back a confirmation. The reason the first one is in the dark is that he never got confirmation. So this is the way to assess this probability that the second general never got the message. And that's greater than 1/2.

So the point is, thinking about that in his head, the first general could decide, I'm not going to venture an attack because with probability greater than 1/2, the other general is not attacking. So then we just formalize the game with a small modification of the environment.

There's two tables here. This is the payoff structure if the enemy is prepared. This is the table structure when the enemy is unprepared-- well, with these probabilities, delta and 1 minus delta. And then there's a two-player game here with two actions each-- to attack or not attack. If they both attack and the enemy is prepared, they both get minus m if the enemy is prepared, or minus m for the general that's attacking and 0 for the other one that is That's the difference. Before, they both suffered here. If you don't attack, you're actually better off with 0 rather than minus m. Of course, if they don't attack, they both get 0.

That's the case, unlikely, but possible that the enemy is prepared. And this is a case that the enemy is unprepared. So now when they both attack, it's a good thing. You have these off-diagonal elements, where one suffers if he attacks and the other one doesn't. And again, if they both don't attack, then you get this neutral 0, 0 outcome.

So in the strategic, meaning they got stuff going on in their head and deciding whether or not to do these actions-- in the strategic coordinated attack problem with the naive communication protocol, it turns out, both generals never attack if the communication system is sufficiently reliable. So this is a disaster.

It is a surprising conclusion if you're not familiar with global games in the sense that epsilon can be very, very small. So almost surely, these messages are arriving. There's a relatively big gain of coordinating a successful attack, a small chance that it will go wrong. And nevertheless, when they're sending these messages back and forth in thinking about what is the likelihood of the other player being in a certain situation or not, the probability-- well, we'll look at this line here.

Suppose the second general never receives a message. The second believes that the enemy is prepared, delta. But it could be the enemy is not prepared. So the overall probability is delta, the enemy is unprepared. With 1 minus delta, the enemy is prepared. And this second general in question here never got the message. So this number is, again, greater than 1/2.

So if second general could feel that way not receiving the message, whatever the first general is doing, the second guy isn't going to attack. And then knowing that, the first general likely would be confused about the reason he wasn't receiving the message from the second one. It's the same. The algebra changes to protect the innocent, but it's the same message every time here. The first general in this case believes that the enemy is unprepared, and the message could be lost, versus the enemy is unprepared. The message was received, but he didn't get the confirmation. And that's a number greater than 1/2.

Well, it turns out, you can just keep going, which is my reference to global games. And that's no matter how many iterations you do. And again, the irony. No matter how many messages are exchanged, which seems to confirm that the enemy is unprepared, it's always a probability greater than 1/2 that something isn't quite right, and you don't execute the attack.

STUDENT: So if you get back to validation questions here, if the second general didn't have a messaging technology and that was common knowledge, then you wouldn't have that issue?

ROBERT M. TOWNSEND: That's the conclusion of the paper.

STUDENT: Right. So for validation, it means you don't want to validate too much.

ROBERT M. TOWNSEND: In this case, right. Either, you don't give the guy the ability to respond, like, a one-way microphone or something, or if he has it, then you need some other commitment to prevent him from sending a validation confirmation message because if the first general gets it in his head that despite the agreement, the guy is going to play it safe and send me a message, then he goes through this deep think. So with commitment to not respond to a message or, better yet, in the technology to never get the message, that worked. That does.

But anyway, the main message here is we need to consider the incentives to follow a protocol. You can't just announce a protocol with good characteristics and necessarily assume that the validators are going to follow it, which, again, is economists' first reaction to the Bitcoin validation. Well, maybe the miners will do something else. Maybe they'll collude. Other things other than following the prescribed Satoshi protocol could potentially happen. And we either have to rule it out or criticize the algorithm.

In mechanism design language, we have the information protocols. Simple or otherwise is one aspect to it. We have the mechanism design aspect, which is relying on messages to implement allocation rules. That combines information design with mechanism design and, in this case, both are being used. So what can I say?

So I think this difference in the word "trust" across economics versus across computer science is an important substantive difference. That doesn't mean that one side is going to be convinced by the other. But when you're thinking about implementing mechanisms on the blockchain, it really matters.

On the one hand, you could say the sacred sauce of the blockchain is that we no longer rely on trusted third parties to do the validation. In the case of Bitcoin as a community currency, not the central bank to provide the currency, we're going to do it in this larger community of mostly well-intended people.

So when people refer-- and this is a substantive difference. When they say blockchain, many people assume that comes with the validation algorithm, that that has to be used. And if it's not being used, and it's not really a decentralized system, then it's not really the blockchain.

My point of view is let's break these things down into pieces. And the smart-contract part can utilize tools of computer science, especially if you-- even if you drop the validation part for how the code works, you still have incentives within layer 2 that are enhanced by the tools of computer science. So we should understand all these components in isolation in terms of what they can do, understand distributed ledgers, understand smart contracts, understand encryption, and think about what we want as a matter of choice.

But likewise, just speaking personally, I'm fine with the notion that you can rely on trusted third parties. It's up to the participants to decide what to trust or not. Where's all this data that's being generated? It's Amazon, Google. And it's not like this vast competitive industry of data storage is going on out there. So most people are fine with it.

There's usually an element of trust somewhere in the system. It's not like it's terrible if that remains. It's more like delineating where it is and deciding if it's OK or if there's a way to do better.

So next time, we'll go through the cryptography, encrypting these messages and in the context of mechanism design. And I'll try to strike a balance between providing you with specific concrete examples of encryption methods. And at the same time, I really can't cover the whole-- or even understand the whole field. So I'll give you some references at the end of the lecture next time for further reading. OK, thank you so much.