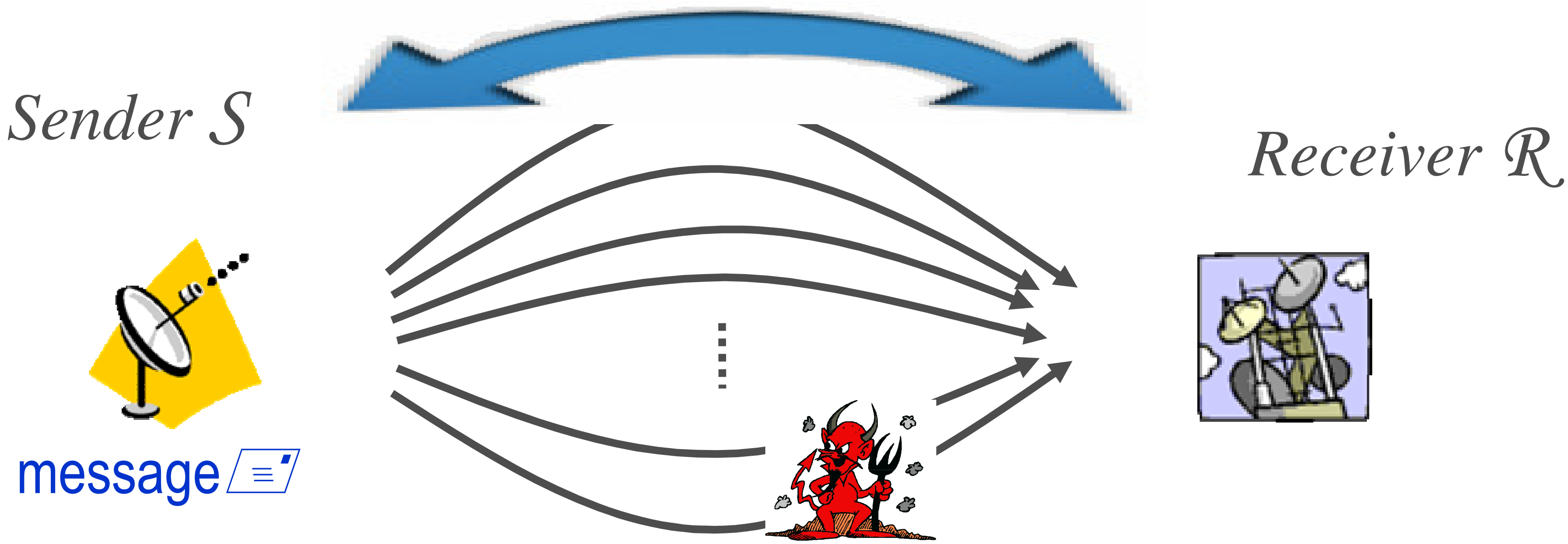
Secure Message Transmission with Small Public Discussion

Juan Garay (AT&T Labs — Research)
Clint Givens (UCLA)
Rafail Ostrovsky (UCLA)

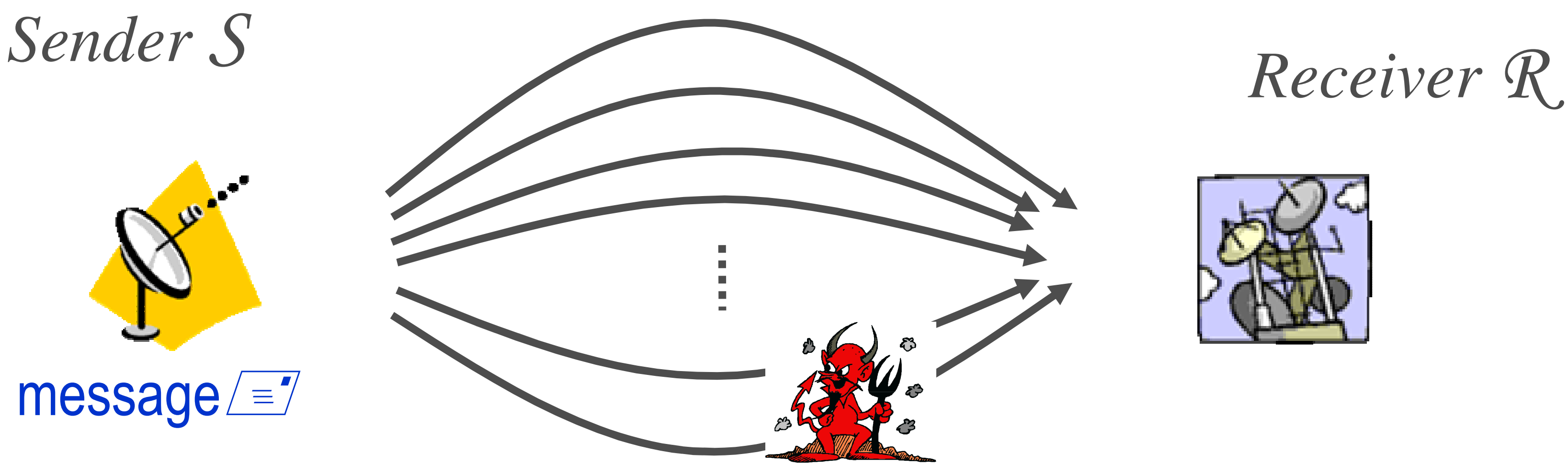
SMT *by Public Discussion* (SMT-PD) [GO08]



Problem: Transmit a message *privately* and *reliably*

- $\mathcal{S}$ and $\mathcal{R}$ connected by ***n*** channels ("wires")
- ***t*** wires (actively) corrupted by adversary $\mathcal{A}$
- … plus an (authentic and reliable) *public channel*

The Original SMT Model… [DDWY93]



Problem: Transmit a message *privately* and *reliably*

- $\mathcal{S}$ and $\mathcal{R}$ connected by ***n*** channels ("wires")
- ***t*** wires (actively) corrupted by adversary $\mathcal{A}$

SMT(-PD): Some Motivation

- *Unconditionally secure* multiparty computation (MPC):

Secure Multi-Party Computation (MPC)

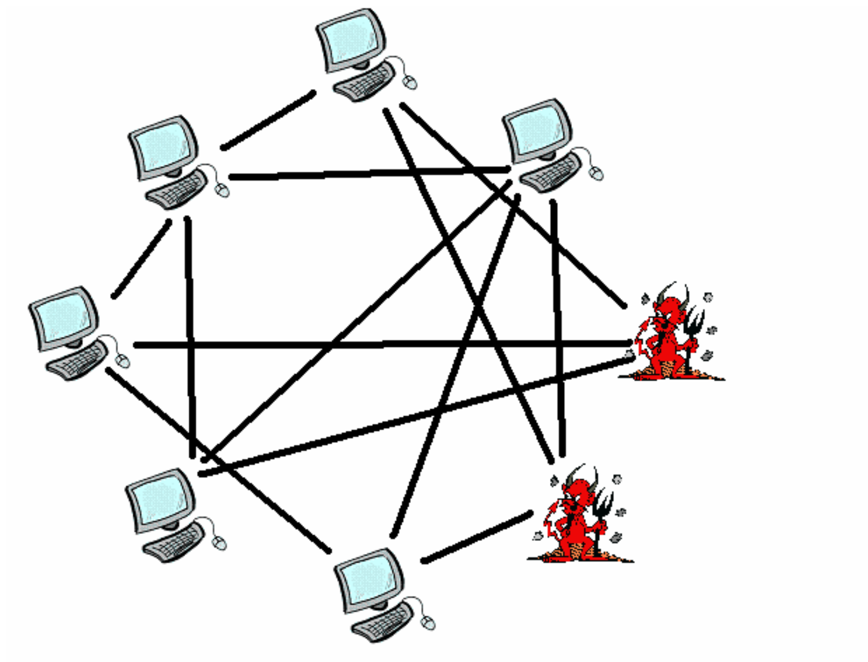
Multi-party computation (MPC) [Goldreich-Micali-Wigderson 87] :

- n parties $\{P_1, P_2, \ldots, P_n\}$, t *corrupted*; each P_i holds a private input x_i
- One public function $f(x_1,x_2,\ldots,x_n)$
- All want to learn $y = f(x_1,x_2,\ldots,x_n)$ (*Correctness*)
- Nobody wants to disclose his private input (*Privacy*)

2-party computation (2PC) [Yao 82] : $n=2$

SMT(-PD): Some Motivation

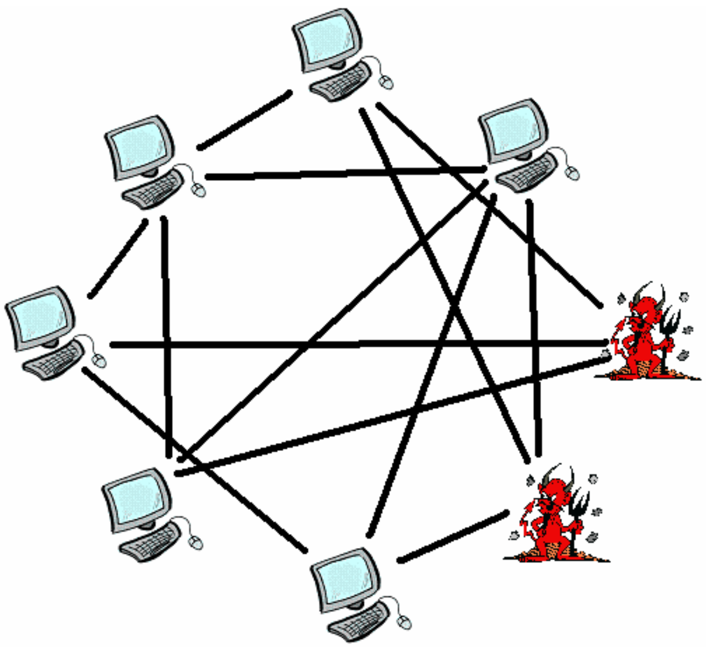
- *Unconditionally secure* multiparty computation (MPC):
  - Possible iff < 1/3 of players are corrupt [BGW'88, CCD'88]
  - Private point-to-point channels sufficient…



… but what if only some of the nodes are connected?

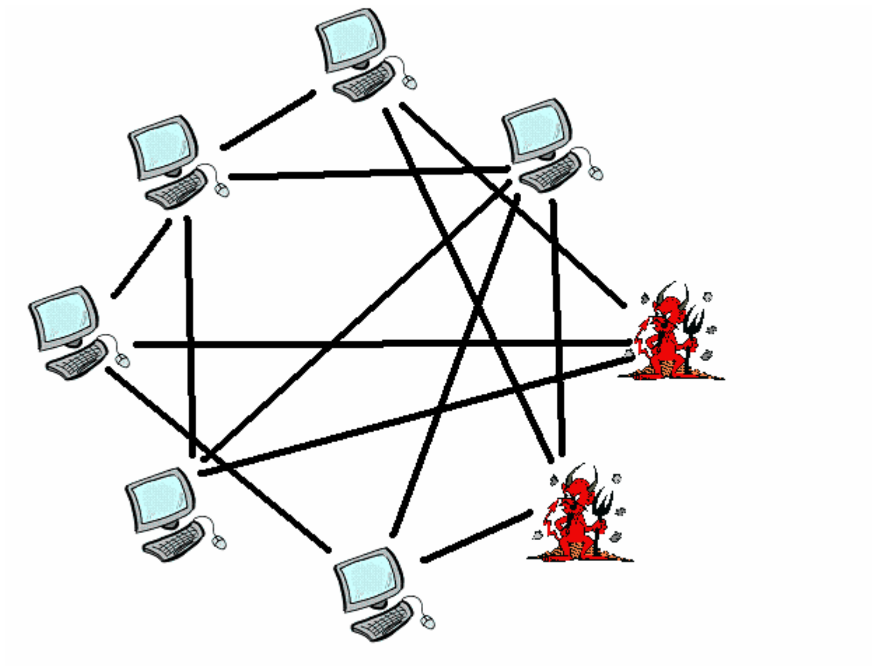
SMT(-PD): Some Motivation

- Idea! [D'82,DDWY'93]: Simulate private p2p channels using SMT protocol



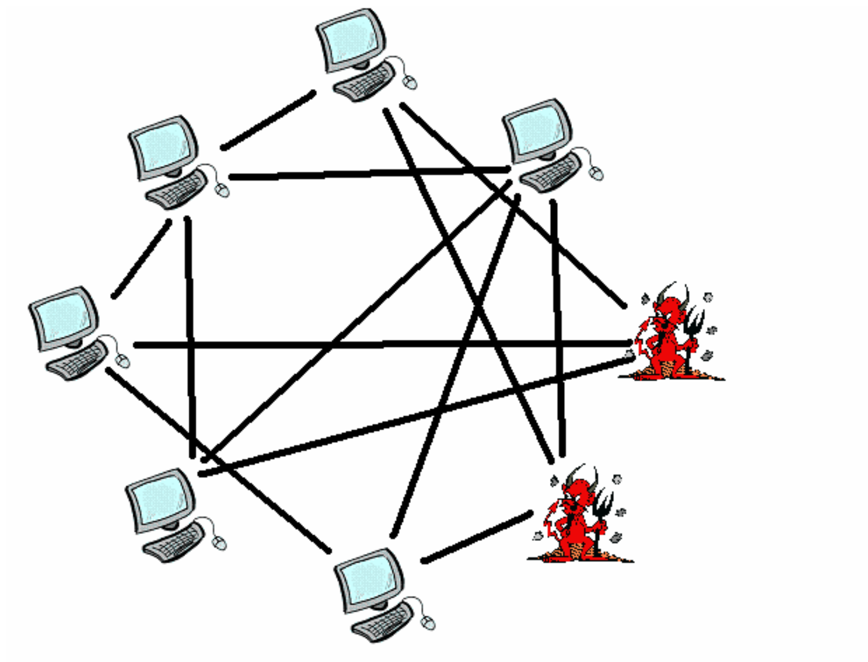
SMT(-PD): Some Motivation

- Idea! [D'82,DDWY'93]: Simulate private p2p channels using SMT protocol
  - Requires connectivity at least 2t+1



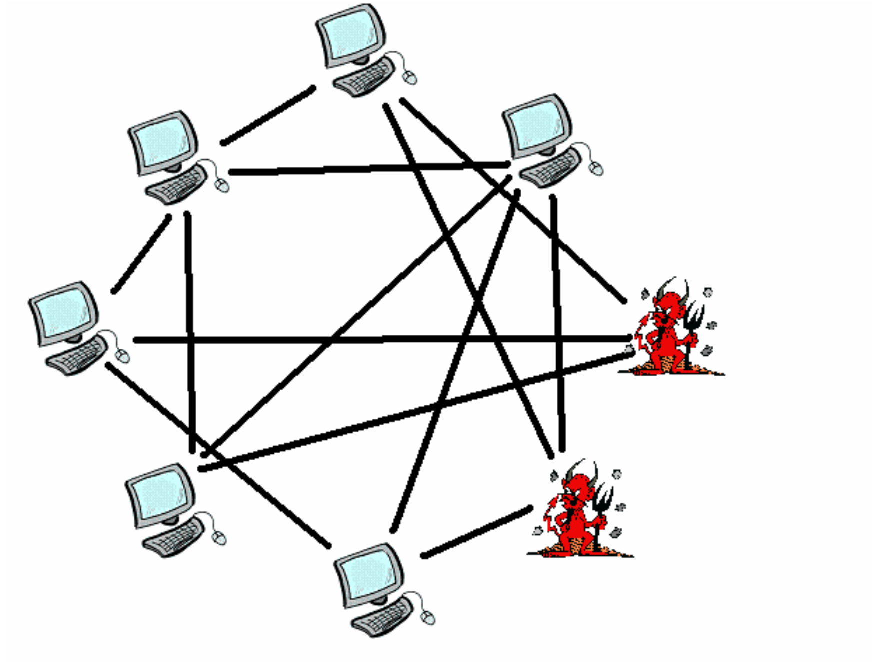
SMT(-PD): Some Motivation

- Idea! [D'82,DDWY'93]: Simulate private p2p channels using SMT protocol
  - Requires connectivity at least 2t+1
  - … Can we do better?



SMT(-PD): Some Motivation

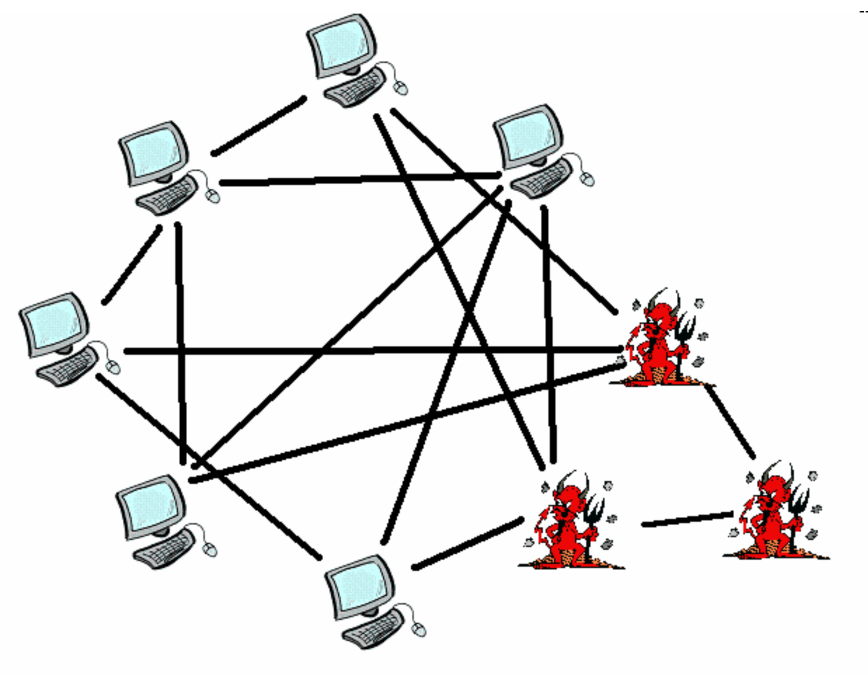
- Idea! [D'82,DDWY'93]: Simulate private p2p channels using SMT protocol
  - Requires connectivity at least 2t+1
  - … Can we do better?



SMT-PD to the Rescue!

- Yes! Can even get **constant** connectivity (!) [GO'08]

  - …but now some of the good guys might be totally cut off from the others…



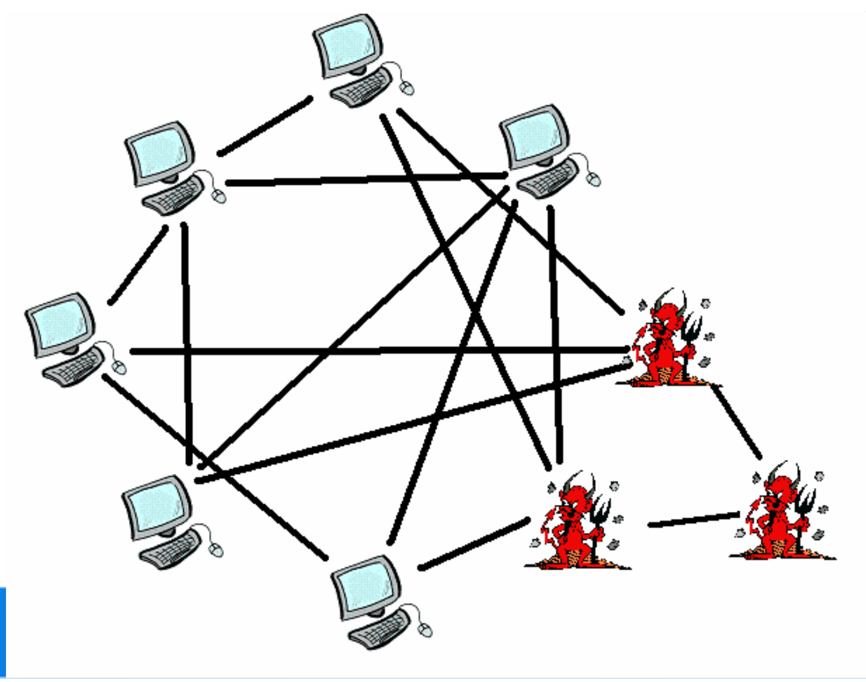
  - So we give up on correctness and privacy for these poor lost souls

smission
cussion

SMT-PD To The Rescue!

Idea! [GO'08]: Simulate private p2p channels using *SMT-PD* protocol

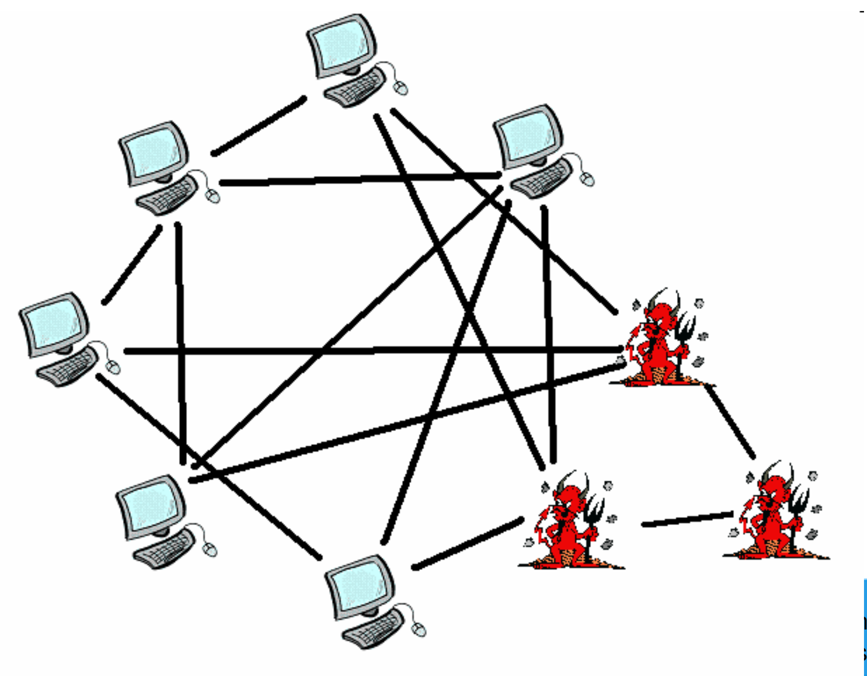
- Possible even for n = t+1 (just one good wire)!



- **The catch:** Must implement a public channel between Sender and Receiver

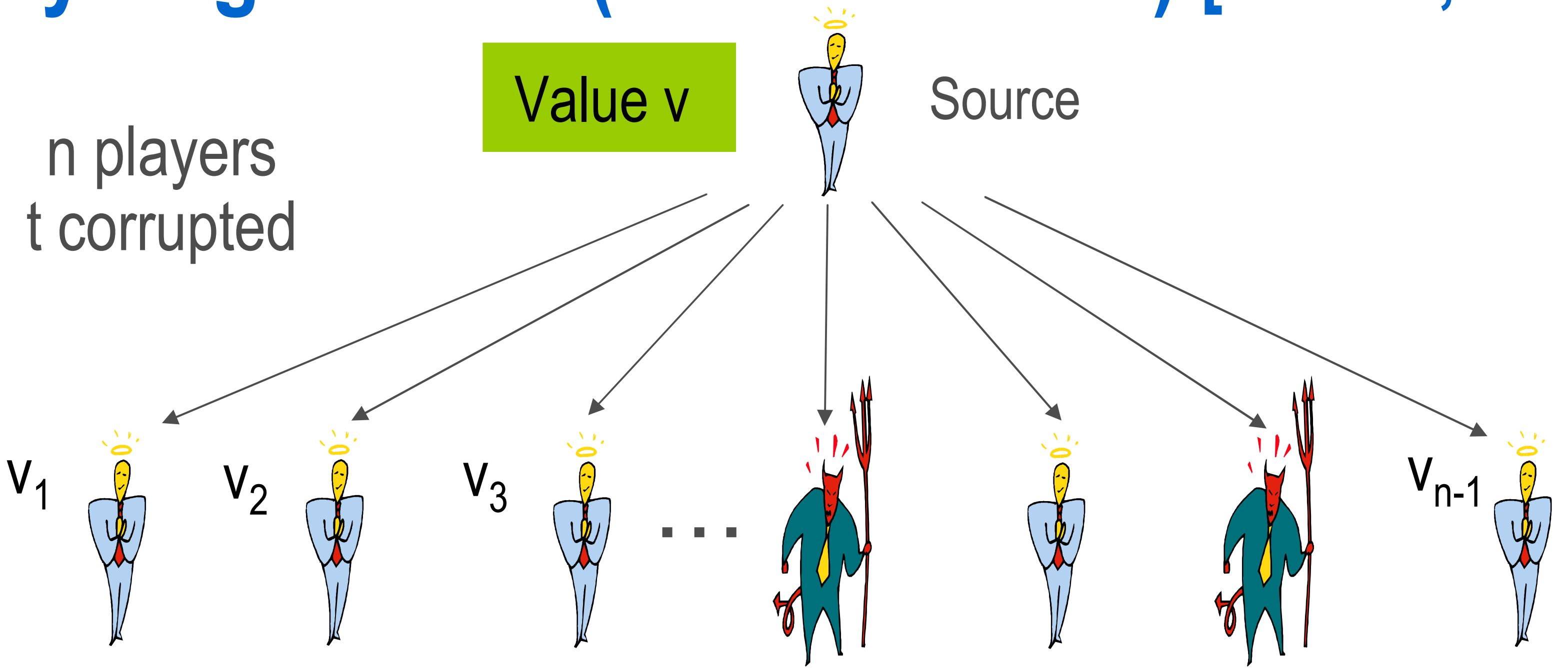
Implementing a Public Channel

- Byzantine agreement for partially connected networks [DPPU'86, Upf'92, BG'93]



Byz. agreement (aka Broadcast) [PSL80, LSP82]

n players
t corrupted

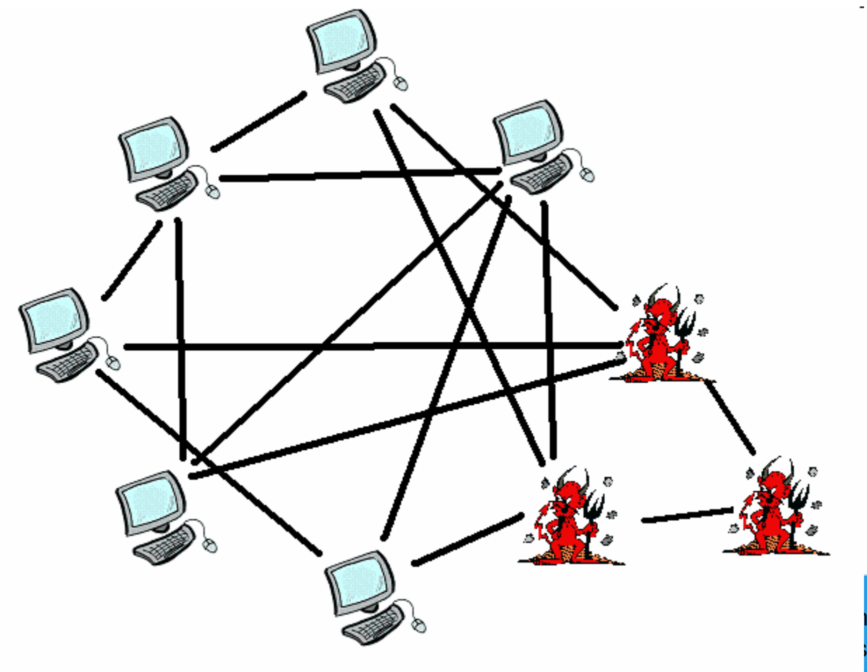


- If source is honest, $v_i = v$ (Validity)
- $v_i = v_j$ (Agreement)

$n > 3t$
(in fully connected networks)

Implementing a Public Channel

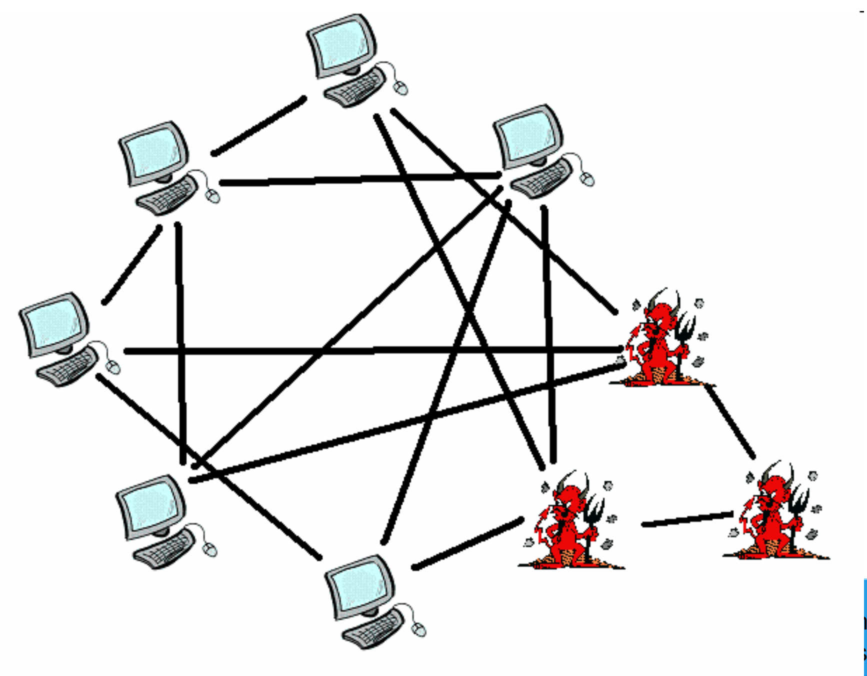
- Byzantine agreement for *partially* connected networks [DPPU'86, Upf'92, BG'93]



ısmission
ːcussion

Implementing a Public Channel

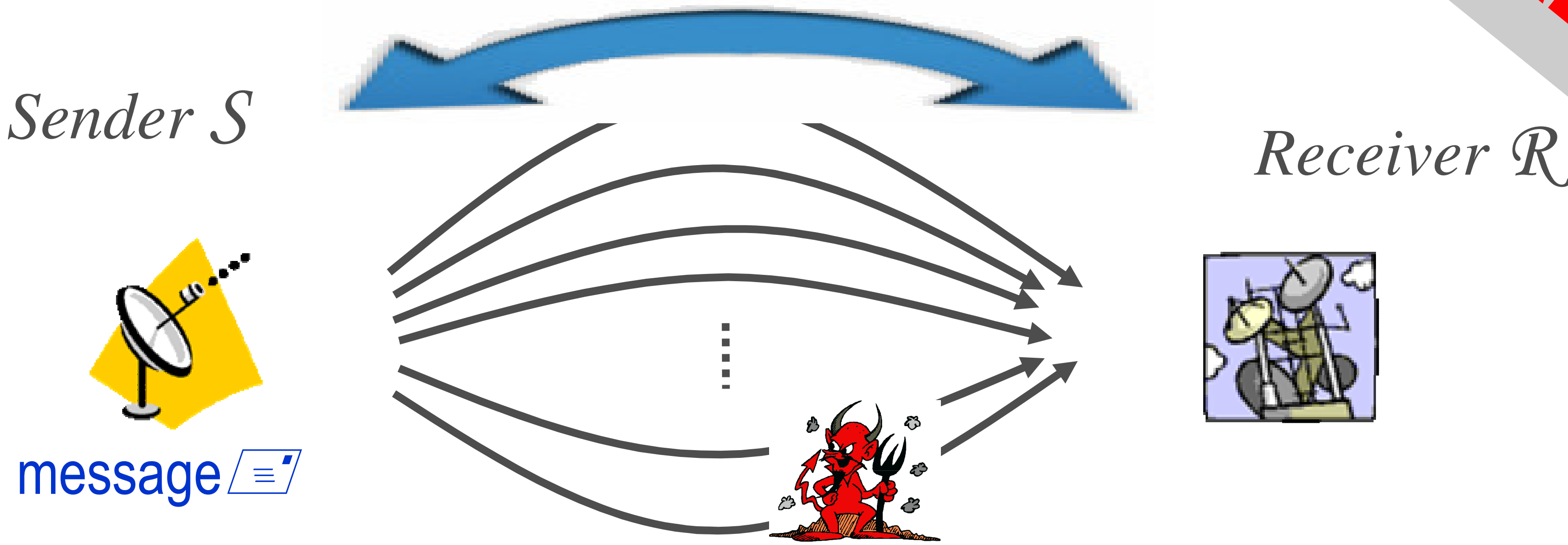
- Byzantine agreement for partially connected networks [DPPU'86, Upf'92, BG'93]
  - This is **EXPENSIVE** in rounds and in communication



- **Question:** Can we minimize use of the public channel in SMT-PD?
- SMT with *Small* Public Discussion

SMT by Public Discussion (SMT-PD) [GO

RECALL



Problem: Transmit a message *privately* and *reliably*

- $\mathcal{S}$ and $\mathcal{R}$ connected by ***n*** channels ("wires")
- *t* wires (actively) corrupted by adversary $\mathcal{A}$
- … plus an (authentic and reliable) *public channel*

A Brief History of SMT

- [Dolev-Dwork-Waarts-Yung'93]
  - *Perfectly* secure message transmission (PSMT)
  - Requires majority of uncorrupted wires, i.e., $n > 2t$
  - 2 rounds necessary, sufficient (in general)
- [Srinathan-Narayanan-PanduRangan'04, Srinathan-Prasad-PanduRangan'07]
  - PSMT comm. complexity = $\Omega(Mn/(n-2t))$
- [Kurosawa-Suzuki'08]
  - PSMT comm. complexity = $O(Mn/(n-2t))$

A Brief History of SMT-PD

- [Franklin-Wright'98] Perfect reliability is *impossible* if majority of wires are corrupt

A Brief History of SMT-PD

- [Franklin-Wright’98] Perfect reliability is *impossible* if majority of wires are corrupt
- [Garay-Ostrovsky’08]
  - 3 rounds, 2 public rounds
  - Public communication = O(Mn)
  - Private communication = O(Mn)

A Brief History of SMT-PD

- [Franklin-Wright'98] Perfect reliability is *impossible* if majority of wires are corrupt
- [Garay-Ostrovsky'08]
  - 3 rounds, 2 public rounds
  - Public communication = O(Mn)
  - Private communication = O(Mn)
- [Shi-Jian-Safavi/Naini-Tuhin'09]
  - 3 rounds, 2 public rounds is *optimal*
  - Public communication = O(M)
  - Private communication = O(Mn)

Previous SMT-PD Protocols Get:

- 3 rounds, 2 public rounds (optimal)
- Perfect privacy, negligible reliability error (optimal)
- Public communication = O(M)
- Private communication = O(Mn)
- **Question: Can we significantly reduce public channel communication?**
- **Question: Can we significantly reduce private wire communication?**

Our Results

- **Upper Bounds**
  - Public communication = $O(n \log M)$
    - previous: $O(M)$
  - Private communication = $O(M\,n/(n-t))$
    - previous: $O(M\,n)$
- **Lower Bounds**
  - Private communication = $\Omega(M\,n/(n-t))$ (matches upper bound!)
- **Amortization**
  - After 2 public rounds, can talk forever

General Structure of SMT-PD Protocol

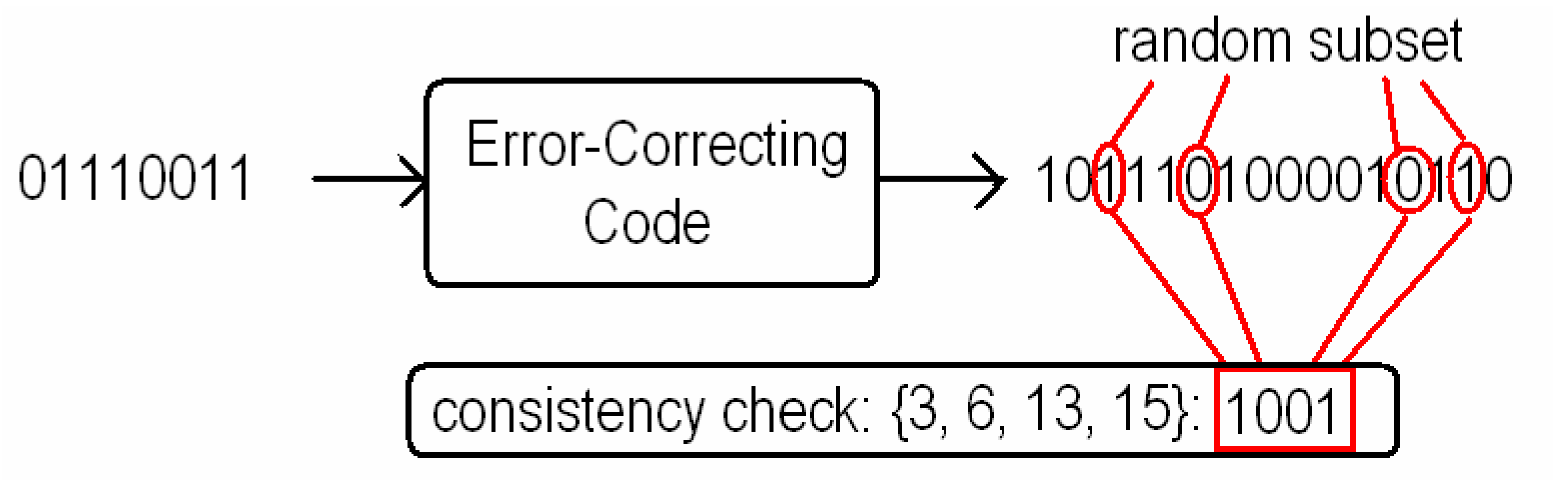
$\mathcal{S}$ wants to send a message to $\mathcal{R}$:

1. $(\mathcal{S} \rightarrow \mathcal{R})$ Send lots of **randomness** over each private wire

2. $(\mathcal{R} \rightarrow \mathcal{S})$ Send **checks** on public channel to verify randomness hasn't been tampered with

3. $(\mathcal{S} \rightarrow \mathcal{R})$ Discard tampered wires. Combine usable randomness into **one-time pad** for message over public channel

Technique: Integrity Checks

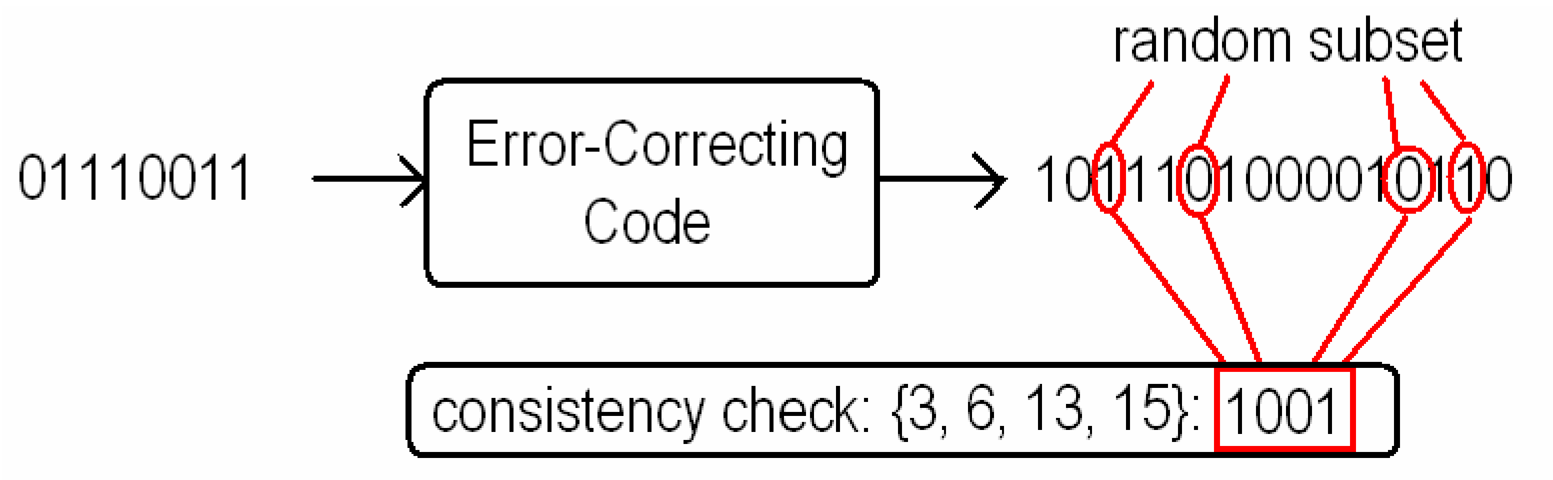


(1) Encode each wire's randomness using an error-correcting code

(2) Reveal small subset of symbols

(3) Reject if received word doesn't match
(or is not a codeword!)

Technique: Integrity Checks



- Suffices to reveal log(n/δ) randomness on each wire
  - δ: reliability error parameter

Fleshing Out the Protocol: Integrity Checks

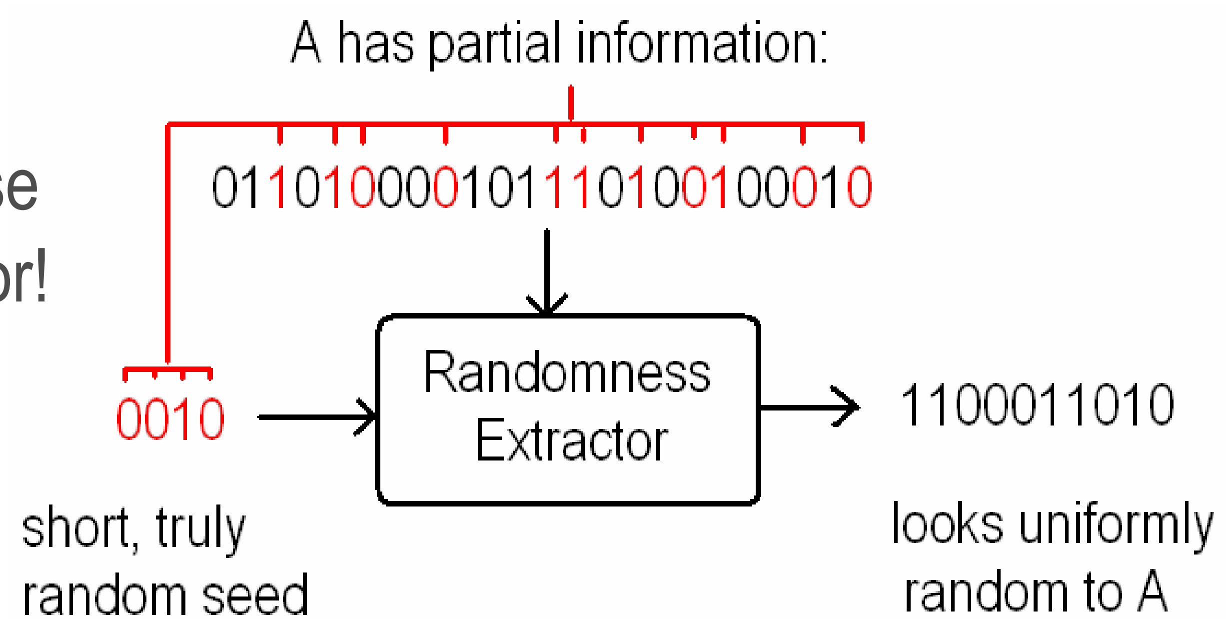
$\mathcal{S}$ wants to send a message to $\mathcal{R}$:

1. $(\mathcal{S} \rightarrow \mathcal{R})$ Send lots of **randomness** over each private wire… *encoded using an Error-Correcting Code*

2. $(\mathcal{R} \rightarrow \mathcal{S})$ Send **checks** on public channel to verify randomness hasn't been tampered with… *by opening a random subset of codeword symbols*

Technique: Hiding the Message

- Previous protocols combine randomness by XOR-ing all usable strings together…
- Have to send O(M) randomness per wire =(
- More efficient: Use randomness extractor! =)



Randomness Extractors

The *min-entropy* of a distribution X over $\{0,1\}^N$ is

$H_\infty(X) = \min_x(-\log \Pr[X = x])$. $H_\infty(X) \geq K \Leftrightarrow \max_x \Pr[X = x] \leq 2^{-K}$.

(X is a "K-source")

Example: Fix N − K of the bits of X, and let the remaining K bits be uniformly, independently random. $H_\infty(X) = K$.

Randomness extractors: Given a sample from *any* source X with sufficient min-entropy, produce an output which is close to uniform.

A function *Ext* : $\{0,1\}^N \times \{0,1\}^s \longrightarrow \{0,1\}^K$ is an $(N, K, k_{min}, \varepsilon)$-strong extractor if

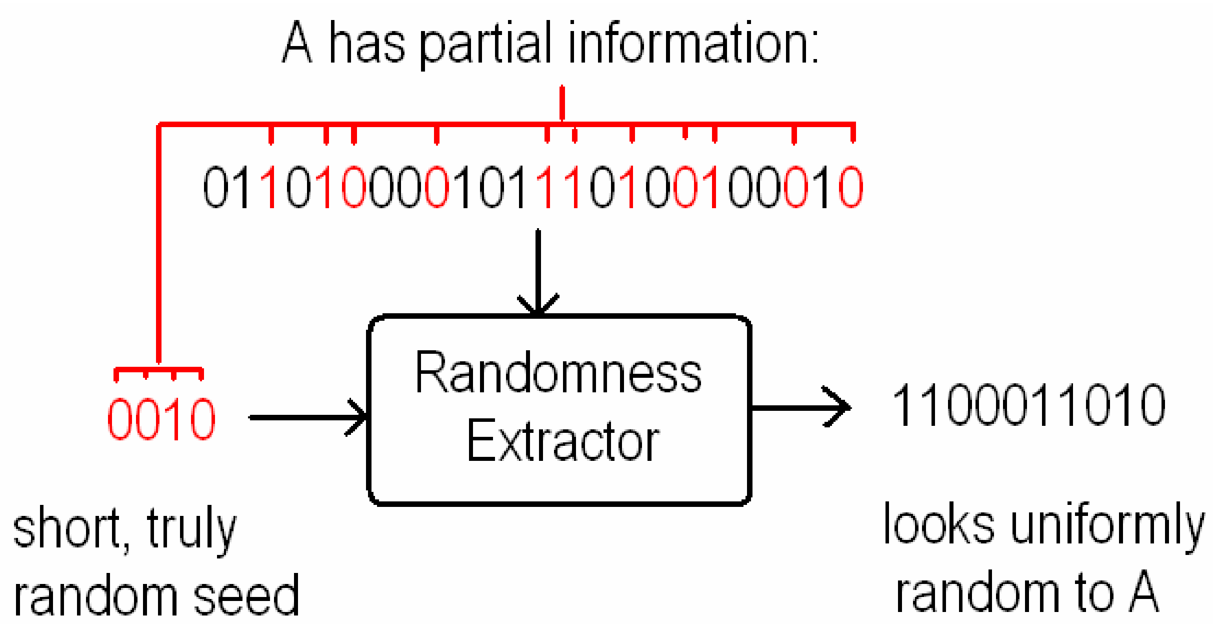
$(U_s, Ext(X, U_s))$ is ε-close to (U_s, U_K) whenever $H_\infty(X) \geq k_{min}$.

at&t

Technique: Hiding the Message (cont'd)

- $\mathcal{A}$ has side information on secret-wire randomness (from Rd 2 integrity checks!)

- Use *average-case* extractor [DORS'04]



Fleshing Out the Protocol: Hiding the Message

$\mathcal{S}$ wants to send a message to $\mathcal{R}$:

2. ($\mathcal{R} \to \mathcal{S}$) Send **checks** on public channel to verify randomness hasn't been tampered with… *by opening a random subset of codeword symbols*

3. ($\mathcal{S} \to \mathcal{R}$) Discard tampered wires. Combine usable randomness… *using an average-case extractor* …into **one-time pad** for message over public channel

What have we gained?

On each **private** wire we can send:

- $O(M / (n-t))$ randomness
- $+ \log(n/\delta)$ extra randomness to account for integrity checks
- $=$ total private-wires communication of $O(Mn / (n-t))$!

(with modest assumptions on M, size of the message)

Now for Public Channel Communication…

2. ($\mathcal{R} \rightarrow \mathcal{S}$) Send **checks** on public channel to verify randomness hasn't been tampered with by opening a random subset of codeword symbols.

- **cheap**: $\Theta(n \log(n/\delta))$

3. ($\mathcal{S} \rightarrow \mathcal{R}$) Discard tampered wires. Combine usable randomness using an average-case extractor into **one-time pad** for message over public channel

- **expensive**: $\Theta(M)$

Now for Public Channel Communication…

3. ($\mathcal{S} \rightarrow \mathcal{R}$) Discard tampered wires. Combine usable randomness using an average-case extractor into **one-time pad** for message over public channel

- **expensive**: $\Theta(M)$

Idea! Why not send the blinded message **over the private wires?**

Yes, Why Not Send It Over Private Wires?

Issue 1: Won't this raise private-wire communication back to O(Mn), thus negating all our hard-fought progress over the last several slides?!

Solution: …Let's think about this later.

Yes, Why Not Send It Over Private Wires?

Issue 2: How will we keep the adversary from tampering with it?

Solution: Let's send a (short!) authentication on the public channel

Issue 3: If we send the authentication at the same time as we send the message (Rd 3), adversary can just choose a tampering consistent with it…?

Solution: Blind the authentication, too

A Short Authentication, Publicly

- For short authenticator, we can use the error-correction integrity checks again:
  - Encode blinded message, send result over each private wire
  - Reveal (logarithmic # of) random symbols on the public channel

A Short Authentication, Publicly

- To hide authenticator, would like a small (size ≈ log M) shared key between $\mathcal{S}$ and $\mathcal{R}$.
  - How to get it?
  - Run a (small) SMT-PD protocol in parallel!

- Since the key is ≈ log M, doesn't hurt us to send it over public channel in Rd 3

Fleshing Out the Protocol: *Parallel* SMT-PDs

$\mathcal{S}$ wants to send a message to $\mathcal{R}$:

1a. ($\mathcal{S} \rightarrow \mathcal{R}$) Send lots of **randomness** over each private wire, encoded using an Error-Correcting Code

- (eventually used to blind message)

1b. ($\mathcal{S} \rightarrow \mathcal{R}$) Send some more ***randomness*** *over each private wire, encoded using an Error-Correcting Code*

- (eventually used to blind authenticator)

Fleshing Out the Protocol: Parallel SMT-PDs

2a. ($\mathcal{R} \rightarrow \mathcal{S}$) Send **checks** on public channel to verify (1a)-randomness hasn't been tampered with, by opening a random subset of codeword symbols

*2b. ($\mathcal{R} \rightarrow \mathcal{S}$) Send **checks** on public channel to verify (1b)-randomness hasn't been tampered with, by opening a random subset of codeword symbols*

Fleshing Out the Protocol: Parallel SMT-PDs

3a. ($\mathcal{S} \rightarrow \mathcal{R}$) Discard tampered wires.

3b. ($\mathcal{S} \rightarrow \mathcal{R}$) Combine usable (1a) randomness using an average-case extractor, into a one-time pad for message over public channel… *Encode (msg+pad) using Error-Correcting Code; send result over every private wire.*

3c. ($\mathcal{S} \rightarrow \mathcal{R}$) Combine usable (1b) randomness using an average-case extractor, into a one-time pad for authenticator…

Construct ***auth*** *by opening ECC(msg+pad) at random subset of symbols; send (auth+pad) on public channel.*

One Last Nagging Question…

Issue 1: Won't this raise private-wire communication back to O(Mn)?!

Solution: Don't send (msg+pad) over *every wire*. (So wasteful!) Instead…

One Last Nagging Question…

- First encode C == (msg+pad) into n shares of size ≈ M/(n-t)
  - Thus, n-t correct shares reconstruct C
- Integrity-check *each share* on public channel
  - Raises Rd. 3 public communication to O(n log M)

Summary: Our Results on SMT-PD

- **Upper Bounds**
  - Public communication = $O(n \log M)$
    - previous: $O(M)$
  - Private communication = $O(M\, n/(n-t))$
    - previous: $O(M\, n)$
- **Lower Bounds**
  - Private communication = $\Omega(M\, n/(n-t))$ (matches upper bound!)
- **Amortization**
  - After 2 public rounds, can talk forever

References

- J. Garay, C. Givens and R. Ostrovsky, "Secure Message Transmission with Small Public Discussion." In *Eurocrypt 2010*. Full paper available from the Cryptology ePrint Archive: eprint.iacr.org/2009/519.
- J. Garay and R. Ostrovsky, "Almost-Everywhere Secure Computation." In *Eurocrypt 2008.*
- N. Chandran, J. Garay and R. Ostrovsky, "Improved Fault Tolerance and Secure Computation on Sparse Networks." In *ICALP 2010.*

Thanks!

Secure Message Transmission with Small Public Discussion

Juan Garay (AT&T Labs — Research)
Clint Givens (UCLA)
Rafail Ostrovsky (UCLA)