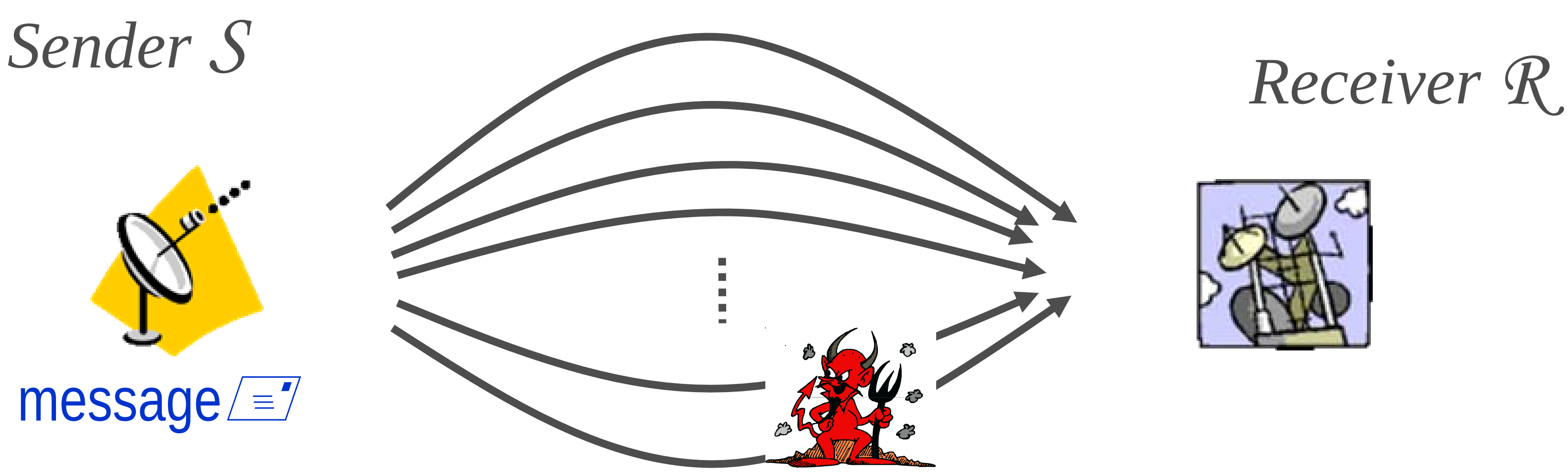
Secure Message Transmission with Small Public Discussion

Juan Garay
AT&T Labs—Research

Clint Givens
UCLA

Rafail Ostrovsky
UCLA

The Original SMT Model [DDWY93]



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

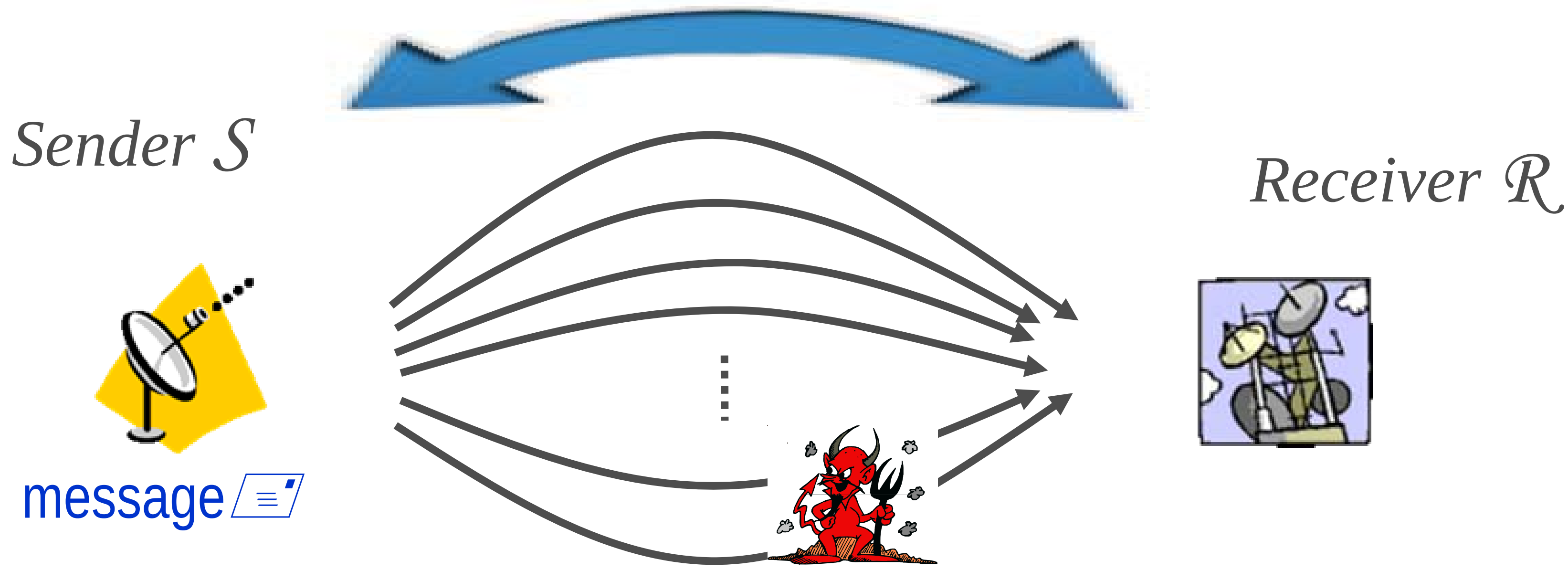
- $\mathcal{S}$ and $\mathcal{R}$ connected by n channels ("wires")
- t wires (actively) corrupted by adversary $\mathcal{A}$ …

An Abridged History of SMT

- [Dolev-Dwork-Waarts-Yung'93]
  - *Perfectly* secure message transmission (PSMT)
  - Requires majority of uncorrupted wires
  - 2 rounds necessary, sufficient (in general)

- [Sayeed-AbuAmara'96, Srinathan-Narayanan-PanduRangan'04, Agarwal-Cramer-deHaan'06, Fitzi-Franklin-Garay-Vardhan'07, Kurosawa-Suzuki'08]
  - PSMT comm. complexity = $\Omega(Mn/(n-2t))$ [SNP'04]

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



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

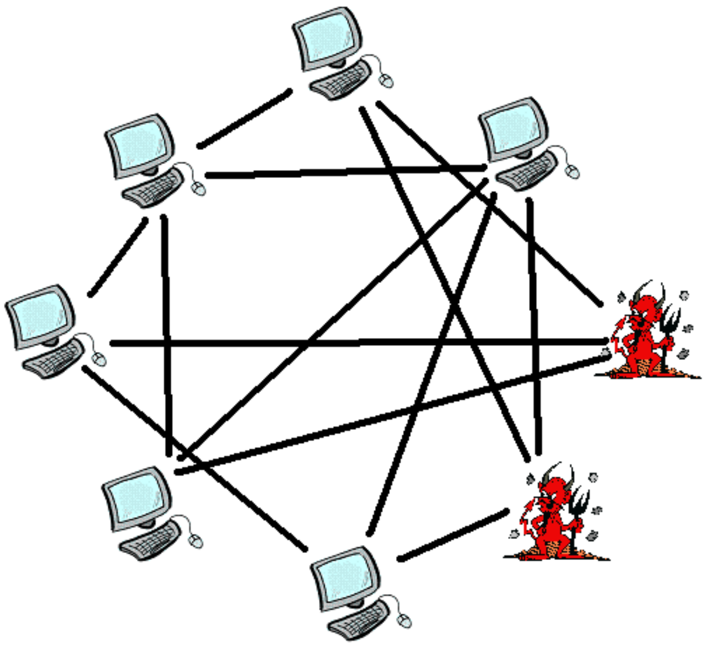
- $\mathcal{S}$ and $\mathcal{R}$ connected by *n* channels ("wires")
- *t* wires (actively) corrupted by adversary $\mathcal{A}$ …
- … plus an (authentic and reliable) *public channel*

A Brief History of SMT-PD

- [Franklin-Wright'98] Perfect reliability is *impossible* if majority of wires are corrupt
- [Garay-Ostrovsky'08] Protocol:
  - 3 rounds, 2 public rounds
  - public communication = O(Mn)
  - private communication = O(Mn)
- [Shi-Jian-SafaviNaini-Tuhin'09]
  - 3 rounds, 2 public rounds is *optimal*
  - public communication O(M)
  - private-wire communication O(Mn)

SMT(-PD): Motivation

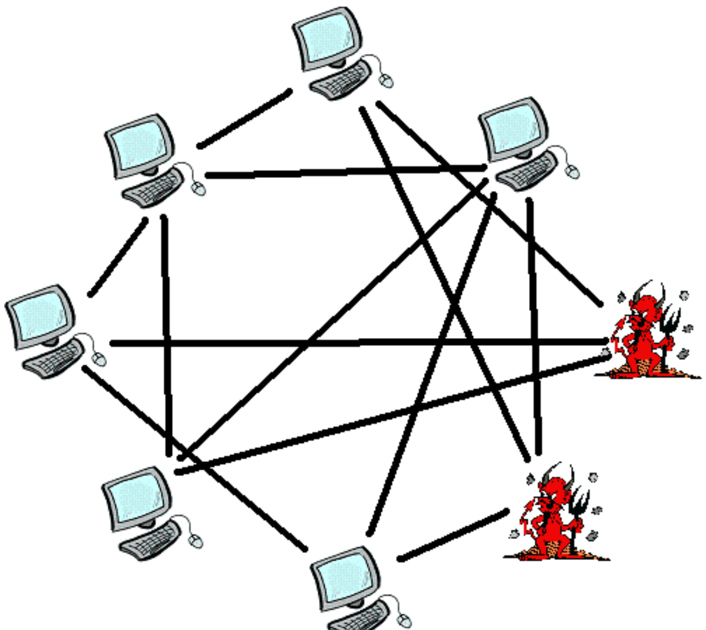
- *Unconditionally secure* multiparty computation:
  - Possible if < 1/3 of players are corrupt [BGW'88, CCD'88]
  - Private point-to-point channels sufficient…



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

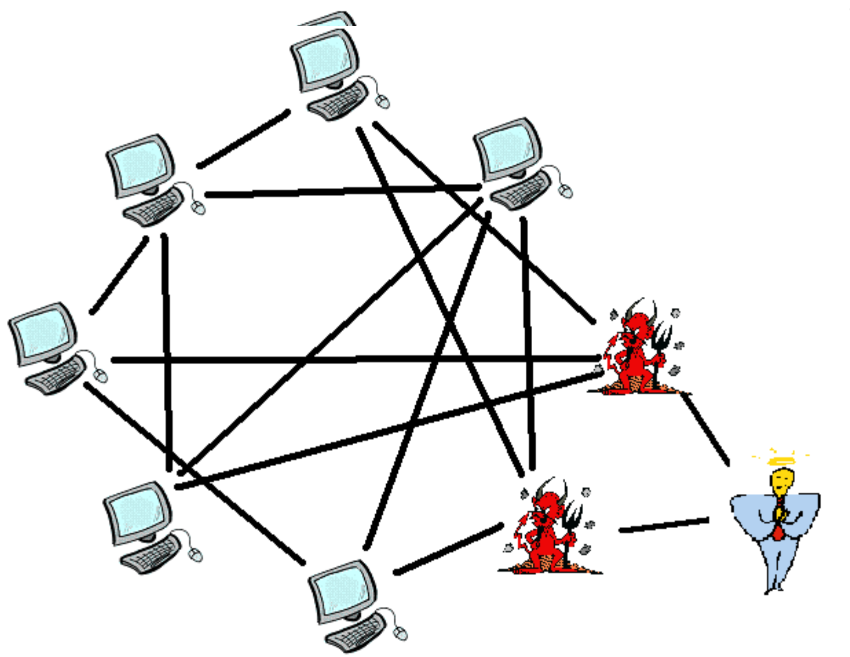
SMT(-PD): Motivation (cont'd)

- Idea! [GO'08]: Simulate private point-to-point channels using SMT protocol
  - SMT requires connectivity at least 2t+1
  - …Can we do better?



SMT-PD To The Rescue!

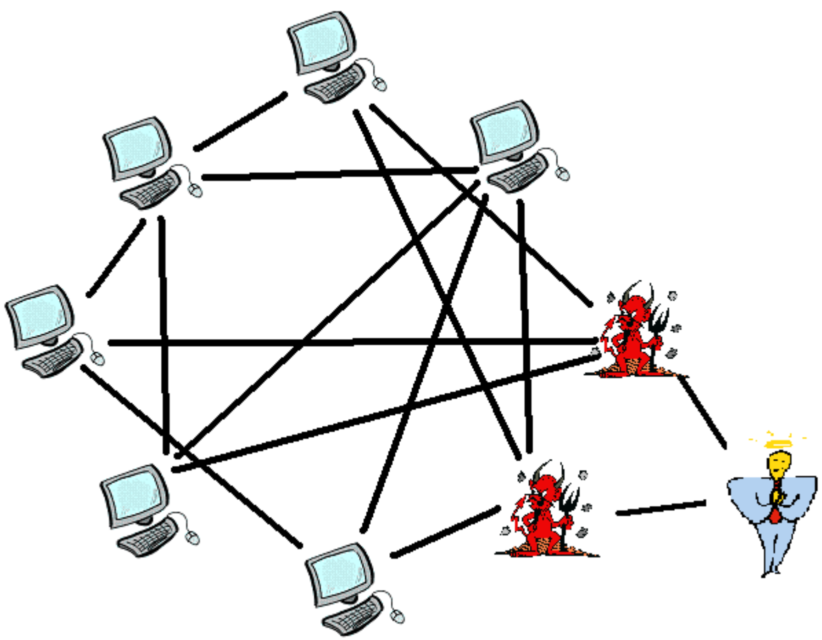
- Yes! Can even get **constant** connectivity (!) [GO'08]
  - …but now some of the good guys might be totally cut off from the others…



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

SMT-PD To The Rescue!

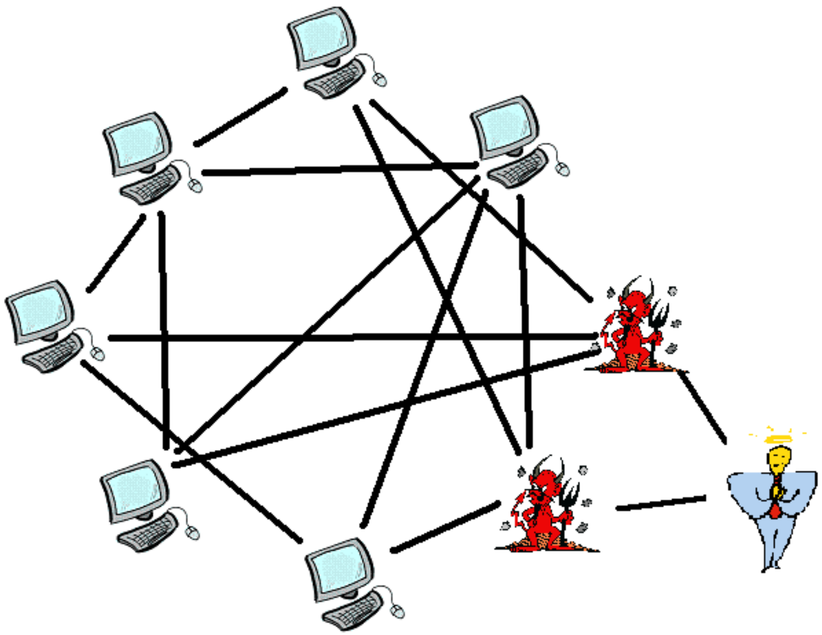
- Idea! [GO'08] Simulate point-to-point connections using SMT-PD protocol
  - Possible even for n = t+1



- **The catch:** Must implement a public channel between Sender and Receiver.
- **Expensive step!**

Implementing a Public Channel

- Broadcast (aka 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?

Previous SMT-PD protocols get:

- 3 rounds, 2 public rounds (optimal [SJST09])
- 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(Mn/(n-t))
    - previous: O(Mn)
- **Lower Bounds**
  - Private communication = Ω(Mn/(n-t)) (matches upper bound!)
- **Amortization**
  - After 2 public rounds, no public rounds needed!

Rest of the talk...

- Explain the upper bound
- For lower bound and amortization, see paper.

General Structure of SMT-PD Protocol

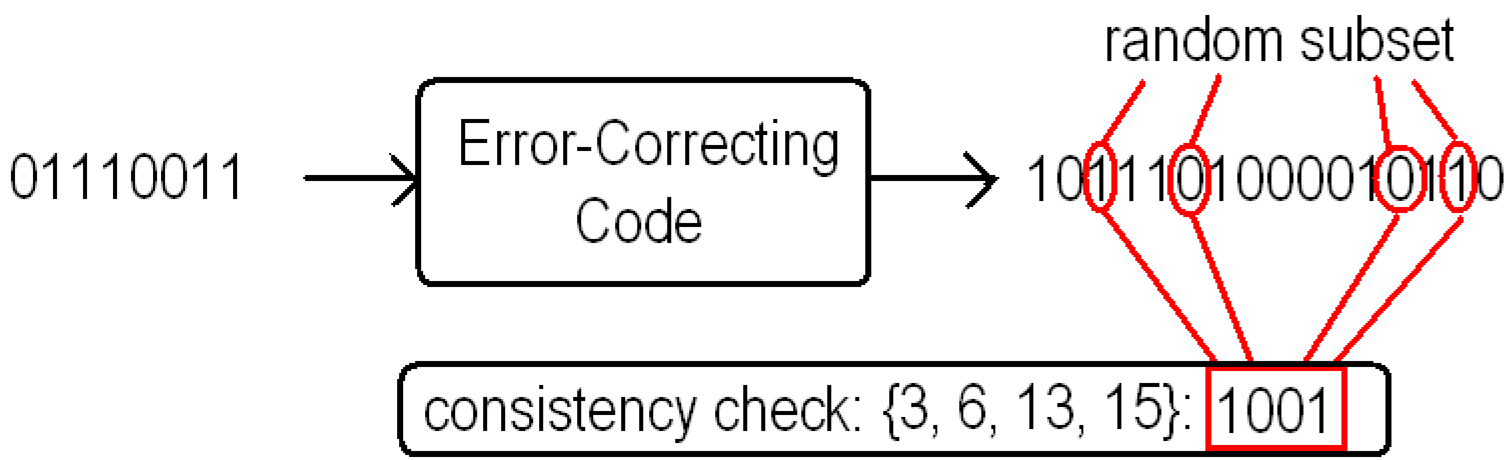
$\mathcal{S}$ wants to send a message to $\mathcal{R}$:

1. ($\mathcal{R} \rightarrow \mathcal{S}$) 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.

Starting point: Simple Integrity Checks

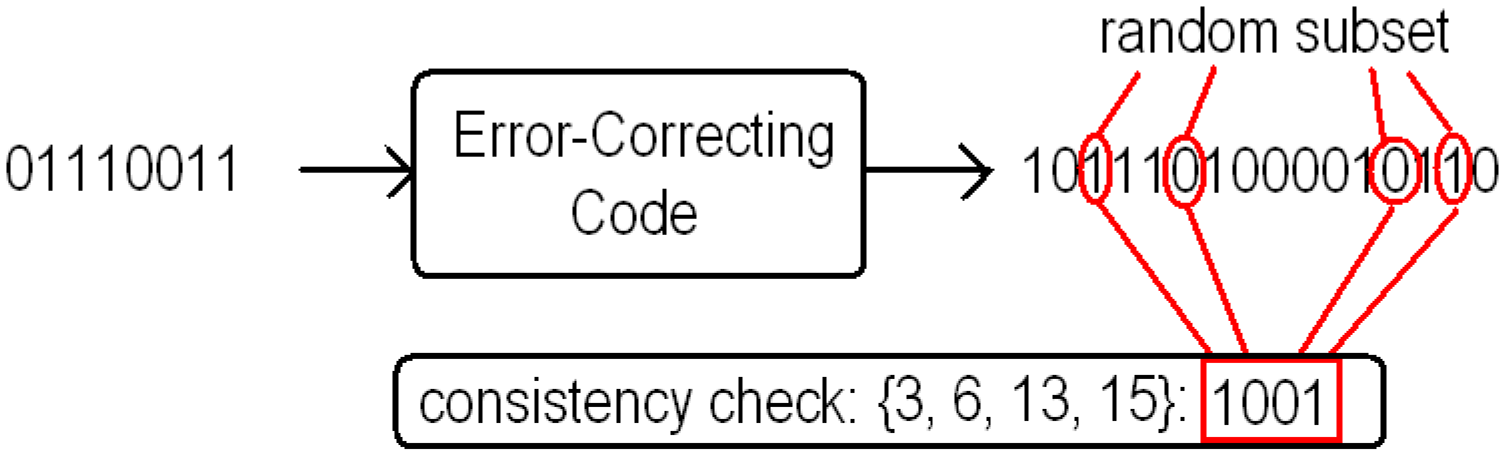


(1) Encode each wire's randomness using an error-correcting code.

(2) Reveal small subset of symbols.

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

What do we get with Integrity Checks?



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

Fleshing Out the Protocol: Integrity Checks

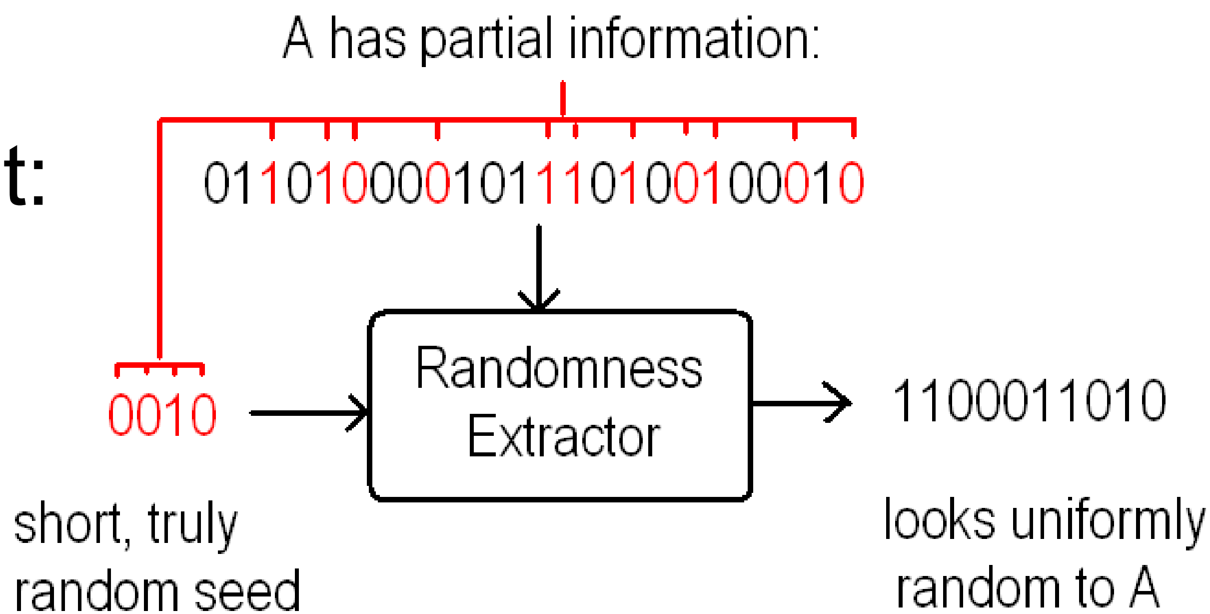
$\mathcal{S}$ wants to send a message to $\mathcal{R}$:

1. ($\mathcal{R} \rightarrow \mathcal{S}$) 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.*

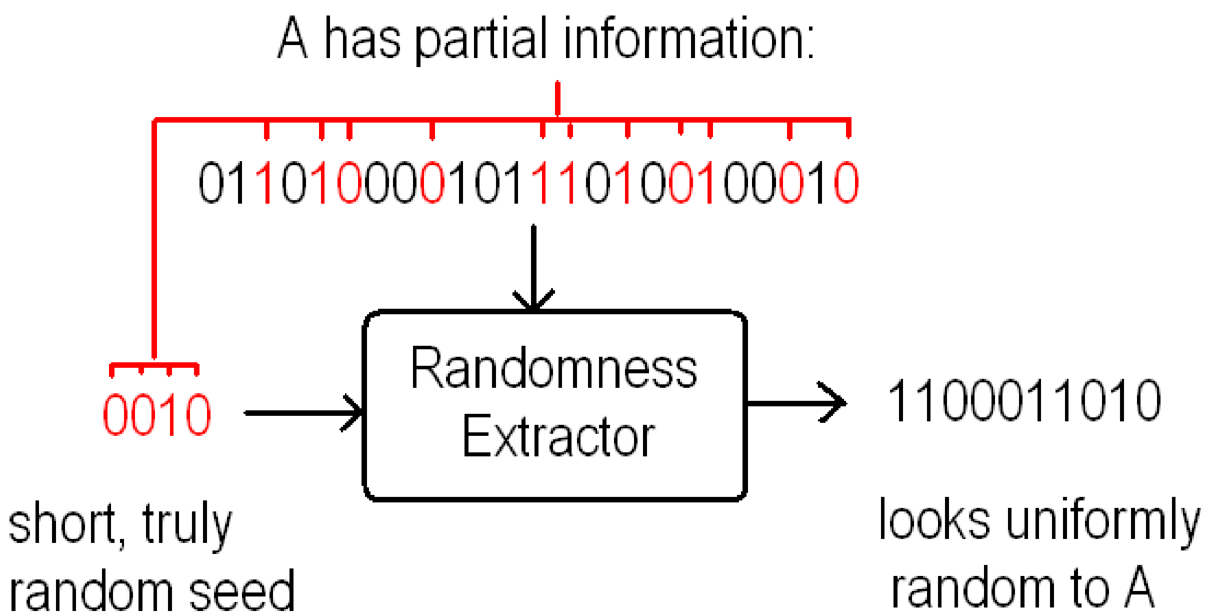
Next Observation: Hiding the Message

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



Next Observation: Hiding the Message

- A has side information on secret-wire randomness (from round 2 integrity checks!)
- Use *average-case* extractor [DORS'04]



Fleshing Out the Protocol: Hiding Message

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

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.*

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.

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 size of M)

Now for Public Channel Communication…

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

- **cheap**: Θ(n log(n/δ))

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

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

- **expensive**: Θ(M)

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.

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 (Round 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 public channel

A Short Authentication, Publicly

- To hide authenticator, would like a small (size $\approx$ log M) shared key between S and R.
  - How to get it?
  - Run a (small) SMT-PD protocol in parallel with the main SMT-PD protocol!
- Since the key is $\approx$ log M, doesn't hurt us to send it over public channel in Round 3

Fleshing Out the Protocol: Parallel SMT-PDs

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

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

- (eventually used to blind message)

1b. ($\mathcal{R} \rightarrow \mathcal{S}$) 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

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

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

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

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.*

Fleshing Out the Protocol: Parallel SMT-PDs

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

*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 $\approx$ M/(n-t).

(so n-t correct shares reconstruct C).

- Integrity-check *each share* on public channel
  - raises Rd. 3 public communication to O(n log M)

Protocol in detail

- $\mathcal{R} \rightarrow \mathcal{S}$: (small) Choose random $r_{i,small}$, $|r_{i,small}| = O(k_{small})$. Send $C_{i,small} = \textit{RS-Enc}(r_{i,small})$ over each wire W_i, $1 \leq i \leq n$.

  (big) Choose random r_i, $|r_i| = O(k)$. Send $C_i = \textit{RS-Enc}(r_i)$ over each wire, W_i, $1 \leq i \leq n$.
- $\mathcal{R} \rightarrow \mathcal{S}$: (small) Open $O(\log(n/\delta))$ randomly chosen positions in $C_{i,small}$, $1 \leq i \leq n$.

  (big) Open $O(\log(n/\delta))$ randomly chosen positions in C_i, $1 \leq i \leq n$.

Protocol in detail (cont'd)

- $\mathcal{S} \rightarrow \mathcal{R}$:

  (small) α_{small} = concatenate $C_{i,small}$ for i non-faulty (pad w/ 0 ∈ $F_{q,small}$).

  Put $W_{sec} = Ext_{q,small}(\alpha_{small})$. ($W_{sec} \in F_q^{r,small} \Rightarrow |W_{sec}| = m_{small}$.)

  (big) α = concatenate C_i for i non-faulty (pad w/0 ∈ F_q).

  Let $C = M + Ext_q(\alpha)$, $C \in F_q^r$.

  Apply RS code $F_q^r \rightarrow F_q^{kn}$: $EncRS(C) = (D_1, D_2, \ldots, D_n) \in F_q^{kn}$.

  View D_i as bit-string of length klog q. Apply binary ECC E':

  $E_i = Enc(D_i)$, $|E_i| = ck \log q$.

  Send E_i on wire W_i (if non-faulty);

  send identities of faulty channels ;

  send $V = W_{sec} \oplus$ {consistency checks for each E_i }.

Protocol in detail (cont'd)

- $\mathcal{S} \rightarrow \mathcal{R}$: (cont'd)

  Receiver : Recover $W_{sec} = Ext_{q,small}(\alpha_{small})$ using non-faulty $C_{i,small}$'s.

  Use V, W_{sec} to get consistency checks for E_i's.

  Interpolate correct E_i's to recover $C = M + Ext_q(\alpha)$.

  Find $Ext_q(\alpha)$ using non-faulty C_i's, subtract to get M.

Conclusions

- SMT-PD with simultaneously:
  - logarithmic (in message size) public communication and
  - optimal private-wire communication
- With an errorless extractor for symbol-fixing sources, we get perfect privacy
- Matching private communication lower bounds
- Save even more public rounds/comm. complexity with amortization

References

Full paper available from the Cryptology ePrint Archive:

eprint.iacr.org/2009/519

Secure Message Transmission with Small Public Discussion

Juan Garay — AT&T Labs—Research

Clint Givens — UCLA

Rafail Ostrovsky — UCLA