SWINBURNE UNIVERSITY OF TECHNOLOGY

Reliable Transmission Over Covert Channels in First Person Shooter Multiplayer Games

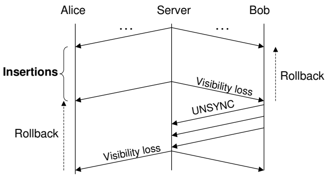
Sebastian Zander,
Grenville Armitage, Philip Branch

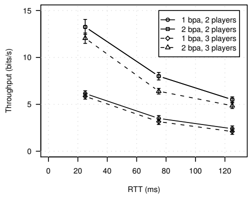
{szander, garmitage, pbranch}@swin.edu.au

Centre for Advanced Internet Architectures (CAIA)
Swinburne University of Technology

Overview

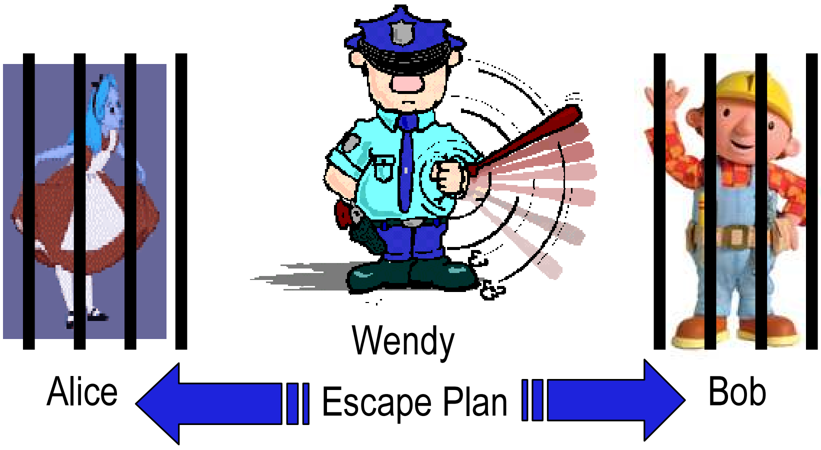
- Covert channels overview
- Covert channels in game traffic
- Channel errors (noise)
- Reliable data transport
- Empirical evaluation
- Conclusions





Often encryption alone is not sufficient

- Encryption protects content of communication
- Existence of communication is enough to take actions
- Covert channels hide existence of communication
- Use means not intended for communication
- Huge amount of traffic in Internet is ideal cover



Wendy

Alice

Escape Plan

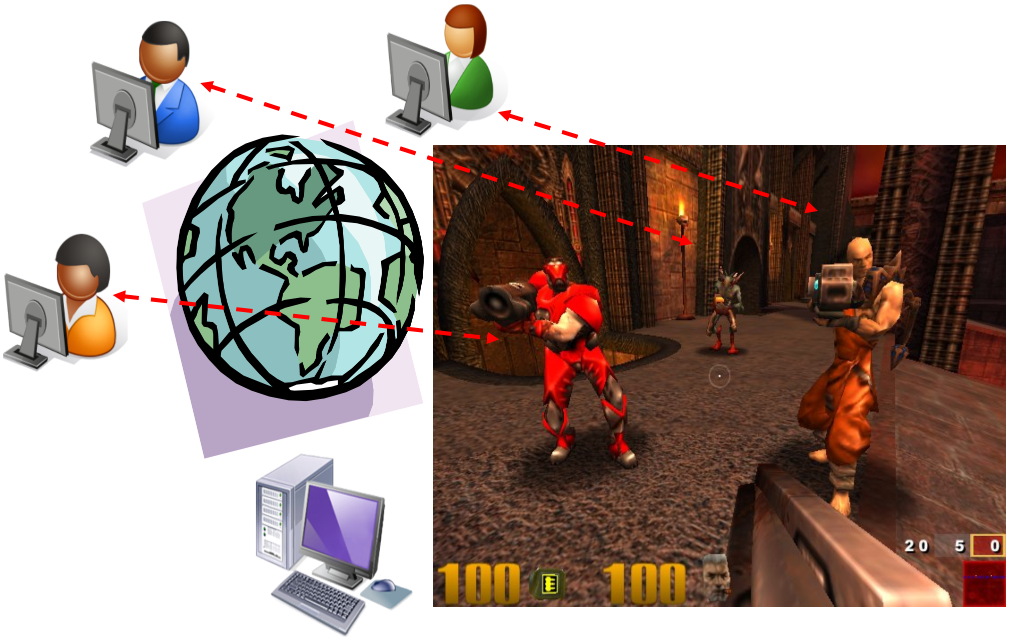
Bob

Covert channels have different users

- Government agencies vs. criminals/terrorists hiding communication
- Hackers ex-filtrating data vs. sysadmins hiding management traffic
- Users circumventing censorship or bypassing firewalls
- Distribution and control of viruses, worms, bots
- Many existing network protocol covert channels
- Very limited work on covert channels in network games (only board games)

Hide covert channels in game traffic

- Hide covert data in variations of player character movements of First Person Shooter (FPS) games
- Channel remains covert if variations are visually imperceptible to players

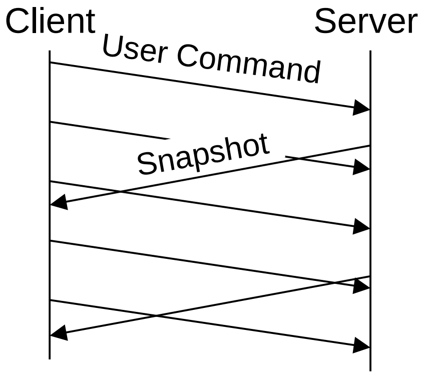


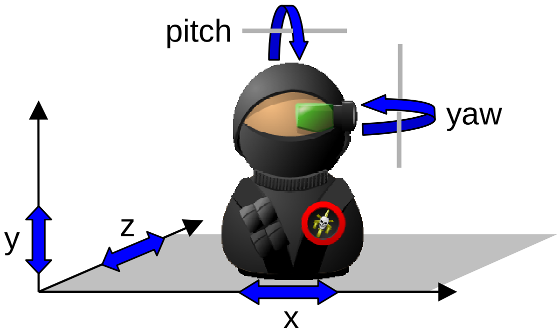
Advantages of FPS covert channels

- FPS games are common and their traffic is not suspicious
- Channel cannot be eliminated because it is tied to player movement
- Sufficient noise in player movement to hide channel
- Sender/receiver use game server as intermediary (tens of thousands active servers)
- Player movements not logged/filtered by servers, unlike in-game chat
- Not limited to FPS → other games, immersive worlds

FPS network protocol overview

- Quake III Arena (Q3) protocol (other games similar)
- Asynchronous message exchange over IP/UDP
- Client sends user commands to server
  - Movement, view angles and buttons
- Server sends game state to client in snapshots
  - State of player character and entities



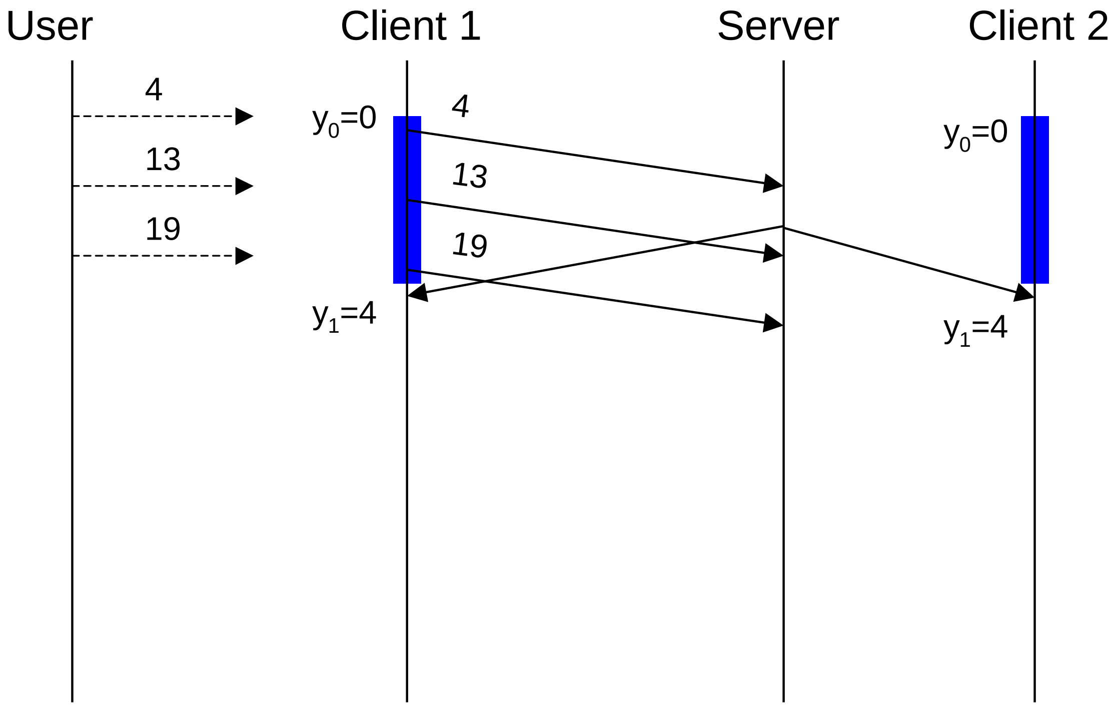


Encoding and decoding of covert bits

- Encode covert data as slight, yet continuous, variations of player character actions
- Encode N covert bits with integer value b in changes of (modified) parameter values $\tilde{y}$ between snapshots:
  $b = |\tilde{y}_j - \tilde{y}_{j-1}| \bmod 2^N$
- Sender can only manipulate $\tilde{y}$ via user commands
  - Use/fire buttons too limited and too obvious
  - Position perturbed by various 'forces'
  - View angles mostly depend on player input only
  - Encode only when player changes angles
- Encode covert bits simultaneously in pitch and yaw

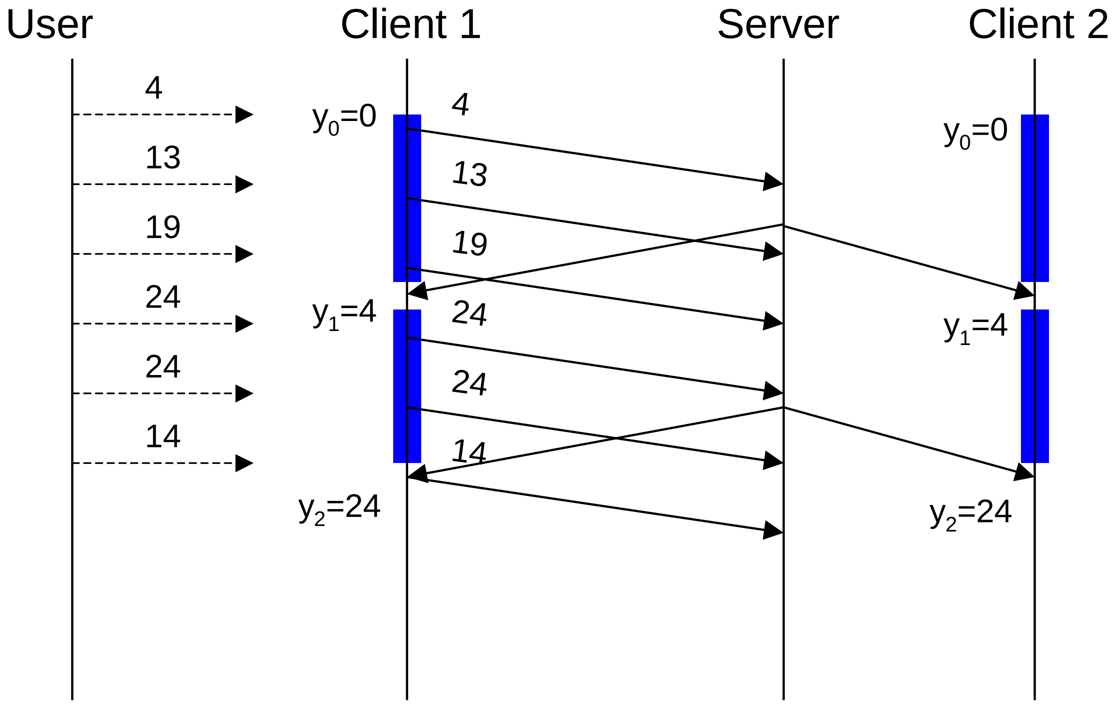
Encoding and decoding example

- Without covert channel



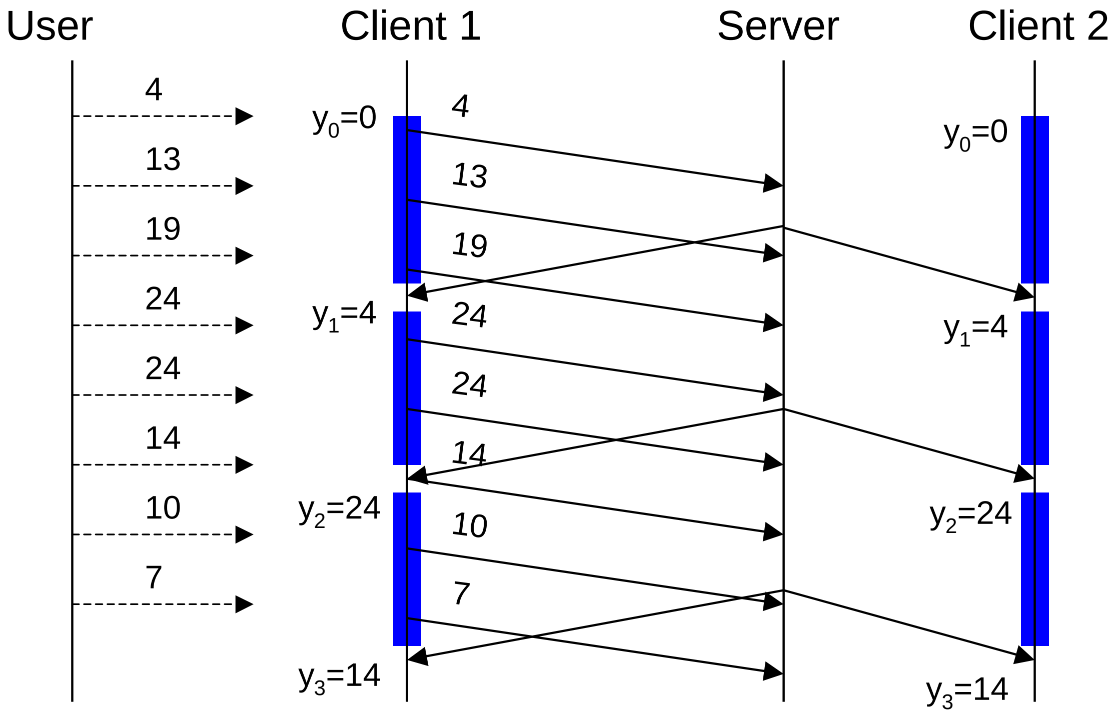
Encoding and decoding example

- Without covert channel



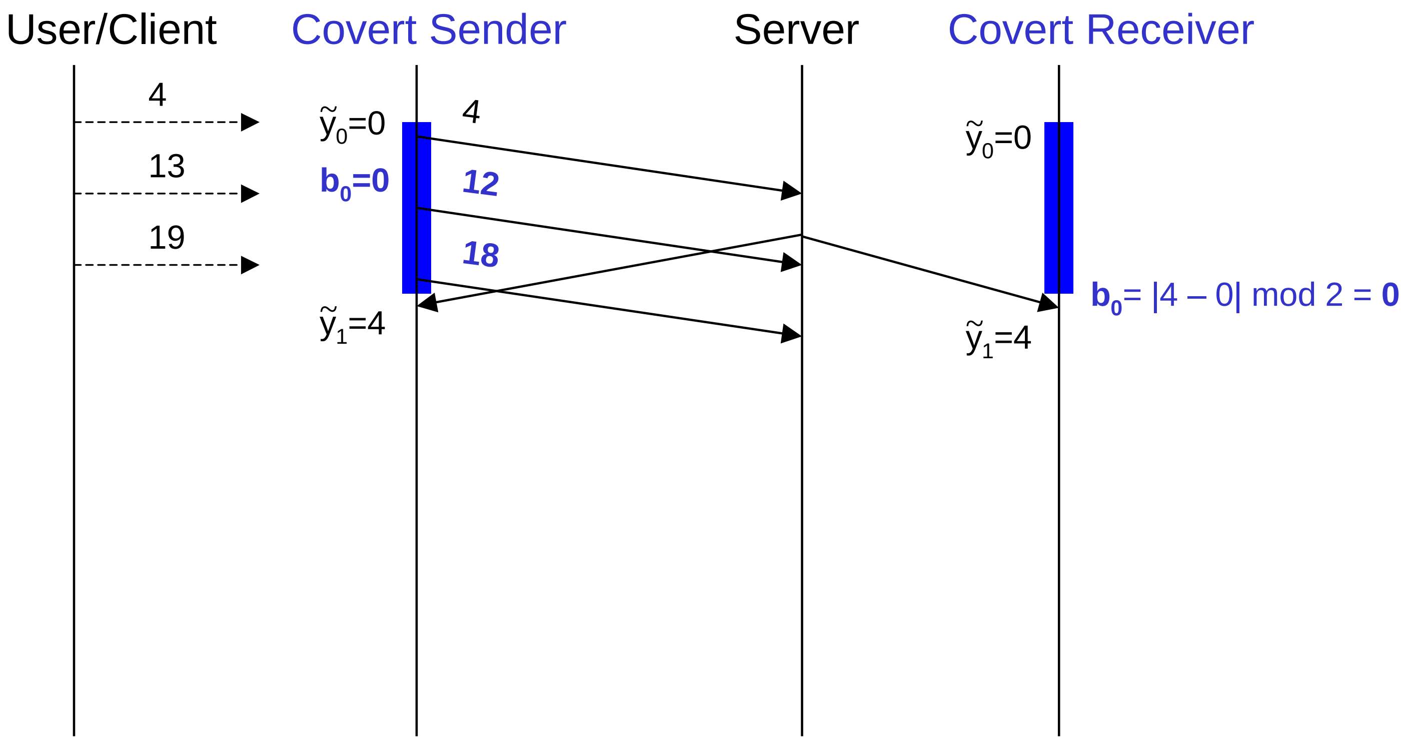
Encoding and decoding example

- Without covert channel



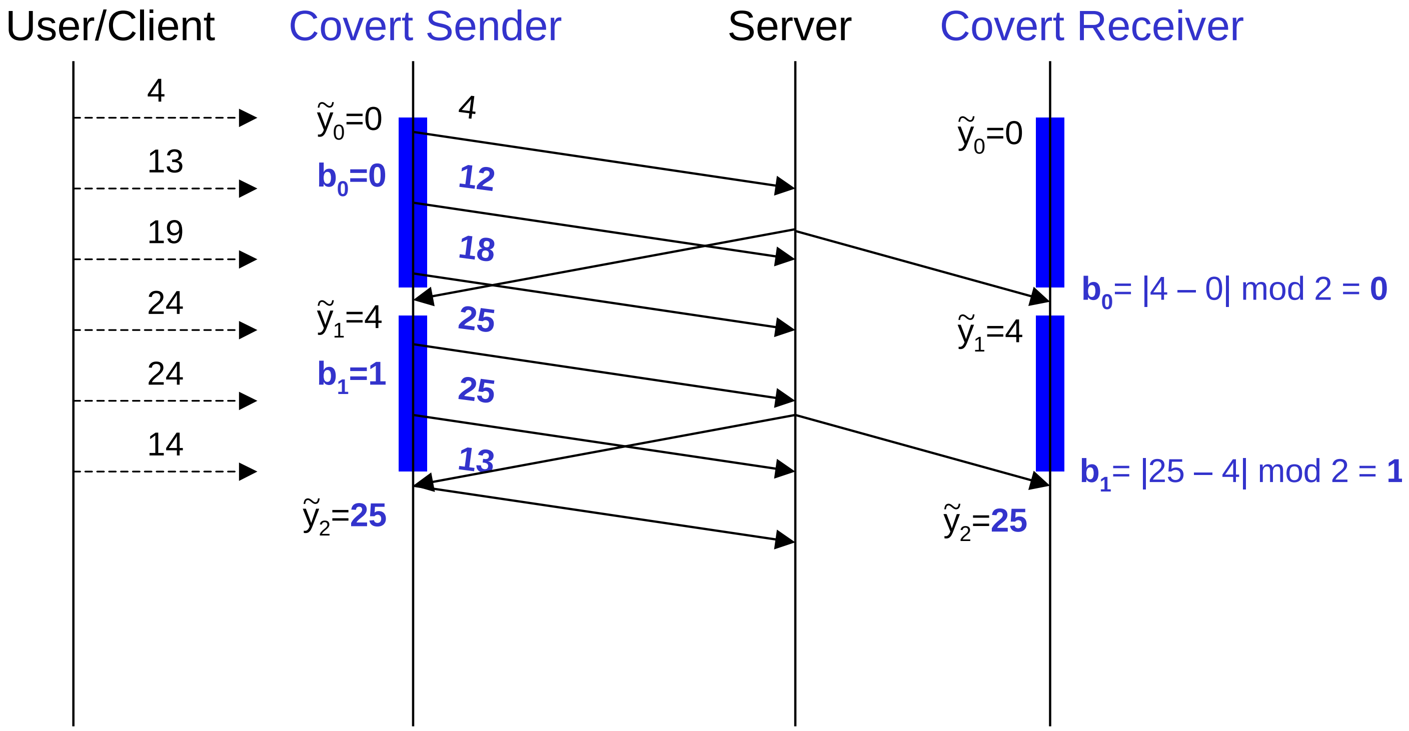
Encoding and decoding example

- With covert channel



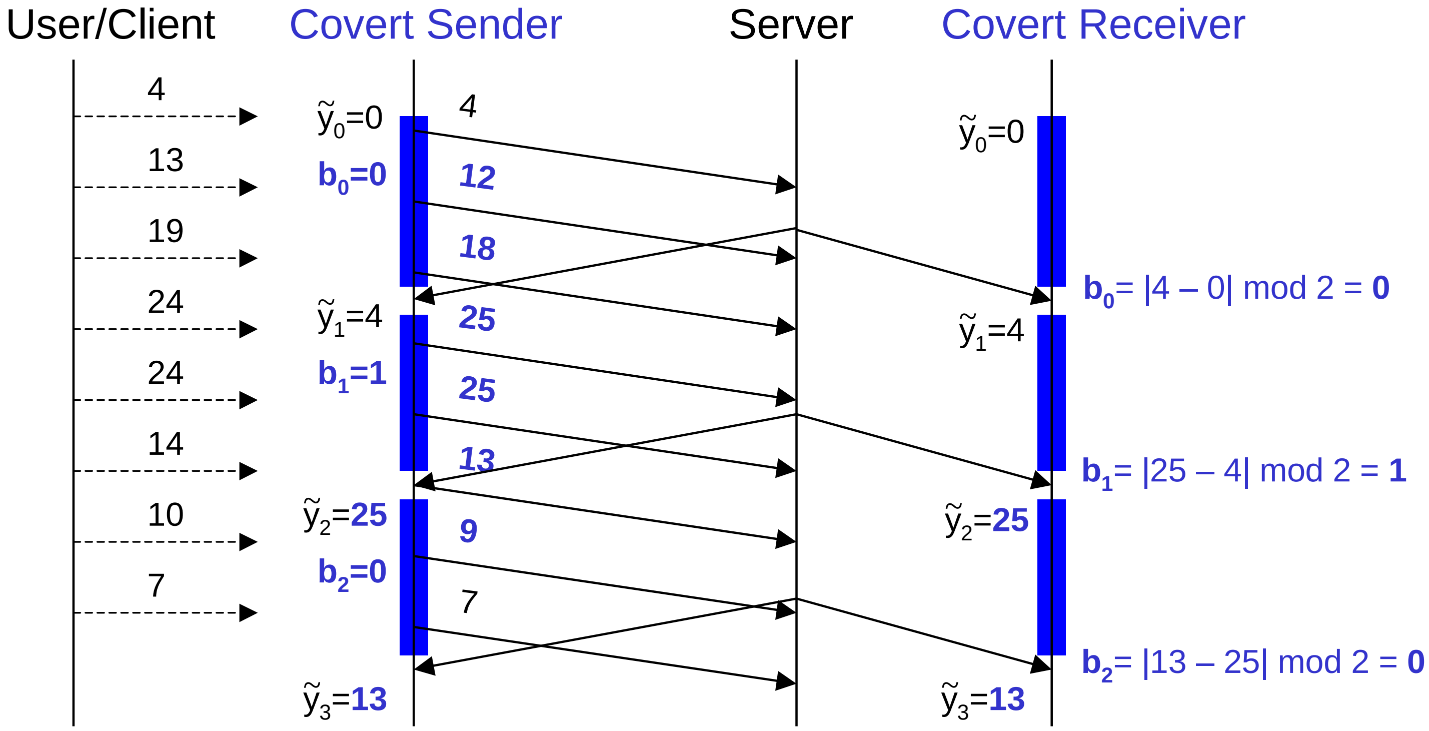
Encoding and decoding example

- With covert channel



Encoding and decoding example

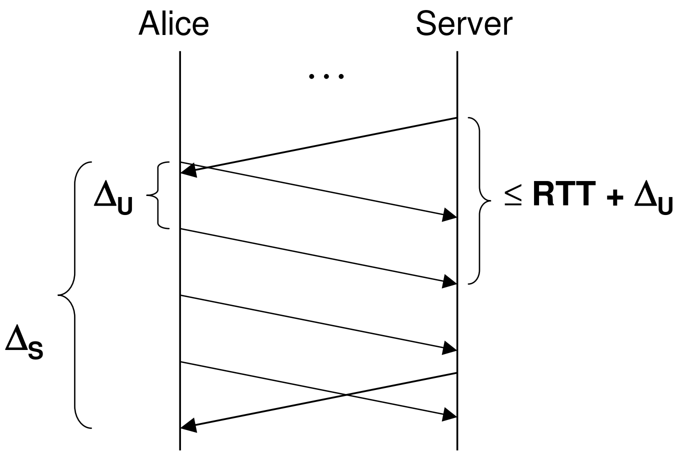
- With covert channel

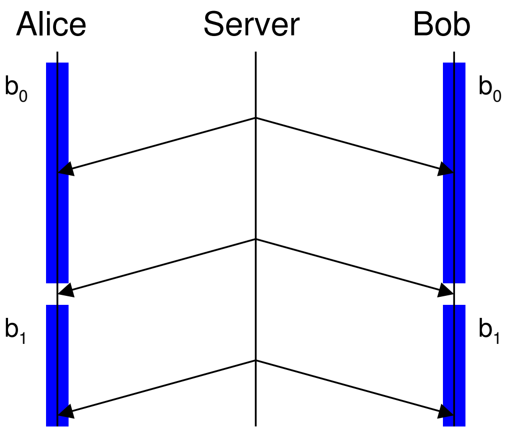


Impact of Round Trip Time (RTT)

- Covert sender encodes bits based on angles from previous snapshot
- RTT must be less than time between snapshots minus time between user commands (typically 40 ms)

⇒ For larger RTTs encode bits in every n-th snapshot

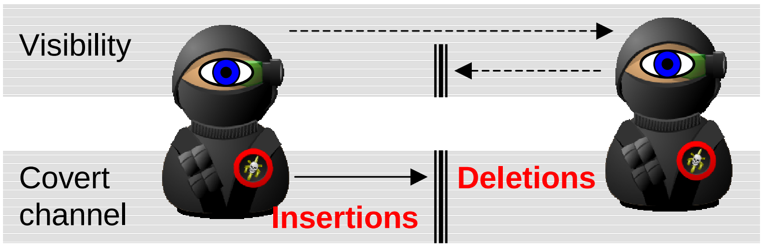




Synchronisation errors

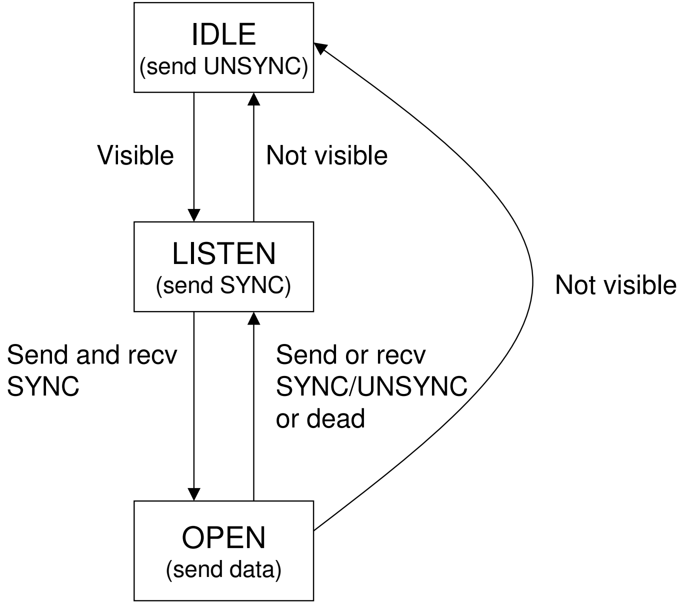
- Synchronisation errors
  - Bits lost on channel (deletions)
  - Bits inserted on channel (insertions)
- Exchange of player state
  - Players only receive state for potentially visible players
  - In Q3 potential visibility is asymmetric
- Lost snapshots (IP/UDP)

⇒ Bit synchronisation mechanism



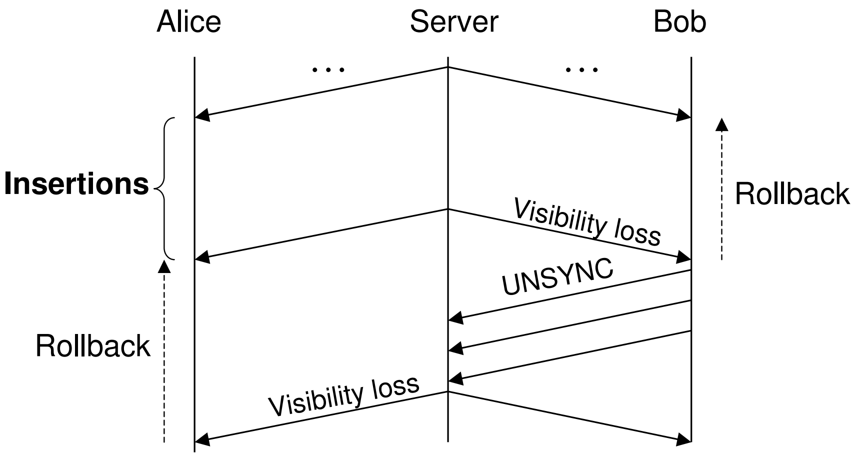
Bit synchronisation mechanism

- A and B let each other know whether they are ready to exchange data
- Use special channel symbols: SYNC, UNSYNC
- Period of data exchange: Transmission Period (*TP*)
- Start of *TP* is synchronised
- End of *TP* is not: B looses visibility to A, A looses visibility to B one snapshot later



Bit synchronisation mechanism cont'd

- Sender
  - Transmit length of data in TP_{i-1} at the start of TP_i
  - Roll back bits send at end of TP_i (→ only insertions)
- Receiver
  - Drop bits inserted in TP_{i-1} based on length info in TP_i
  - Drop bits of incomplete bytes (byte synchronisation)

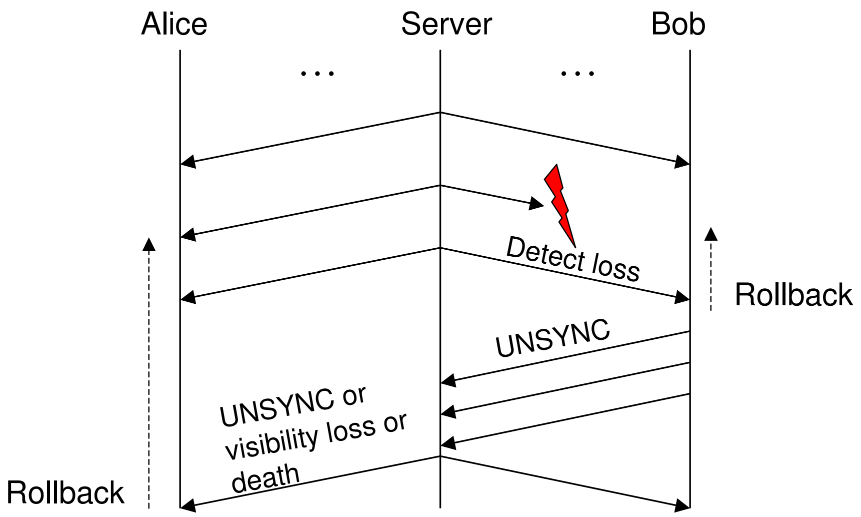


Bit synchronisation mechanism cont’d

- Detect lost snapshots using Q3 sequence numbers

⇒ End transmission period

- B knows number of snapshots lost, but cannot tell A
- Number of bits to roll back must be pre-configured for longest possible loss burst



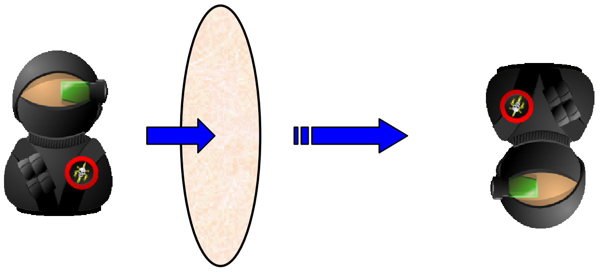
Substitution errors

- Substitution errors = flipped bits
- Teleportation including respawning after death
- Lost user commands (IP/UDP)
- Moving platforms

⇒ End transmission period

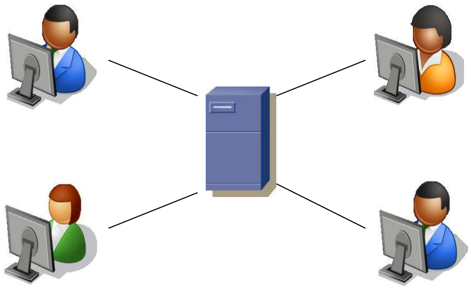
- Pitch clamping

⇒ Pause encoding and decoding



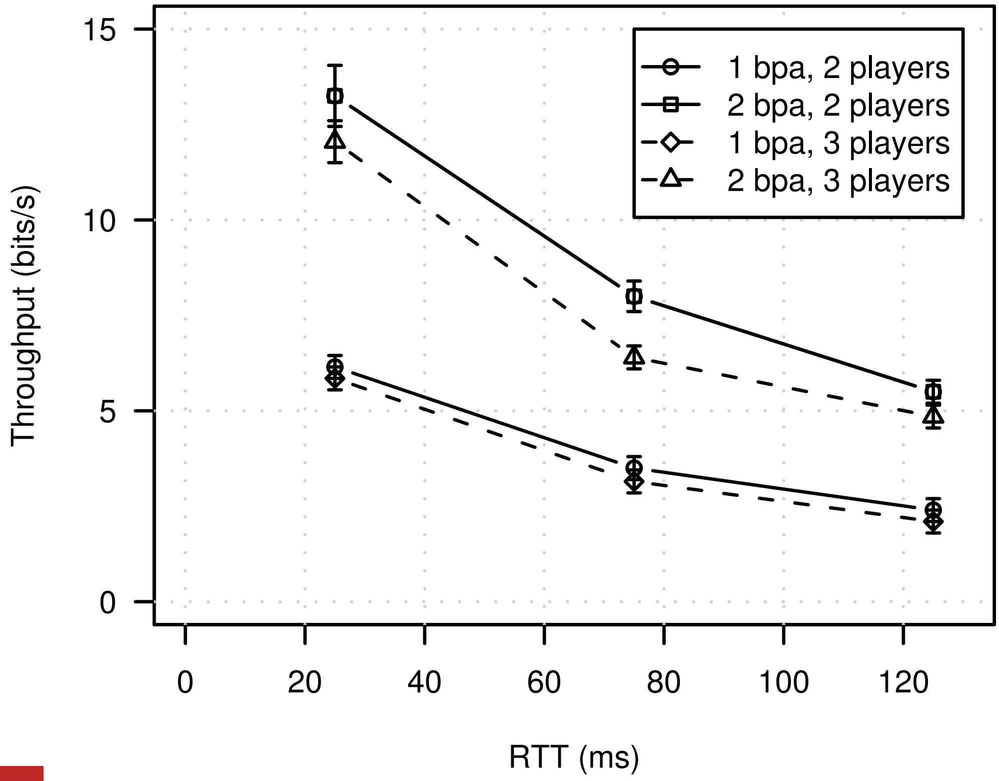
Evaluation in local testbed

- One Q3 server and 2–3 Q3 clients
- Covert sender/receiver are transparent proxies
- Players
  - Client-side bots → don't change behaviour or get tired
  - Limited tests with human players
- Five one-hour games per parameter setting
- Emulate packet delay and loss (Linux Netem)
- Measure average throughput



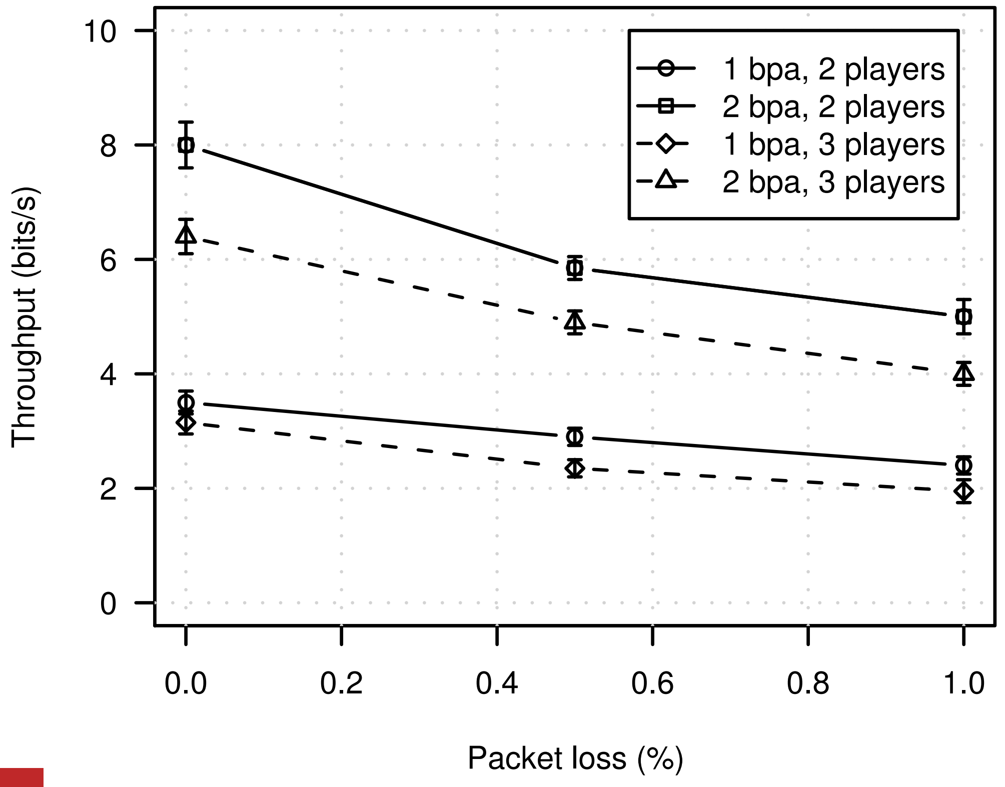
Throughput depending on RTT

- 25 ms, 75 ms, 125 ms RTT (0% packet loss)



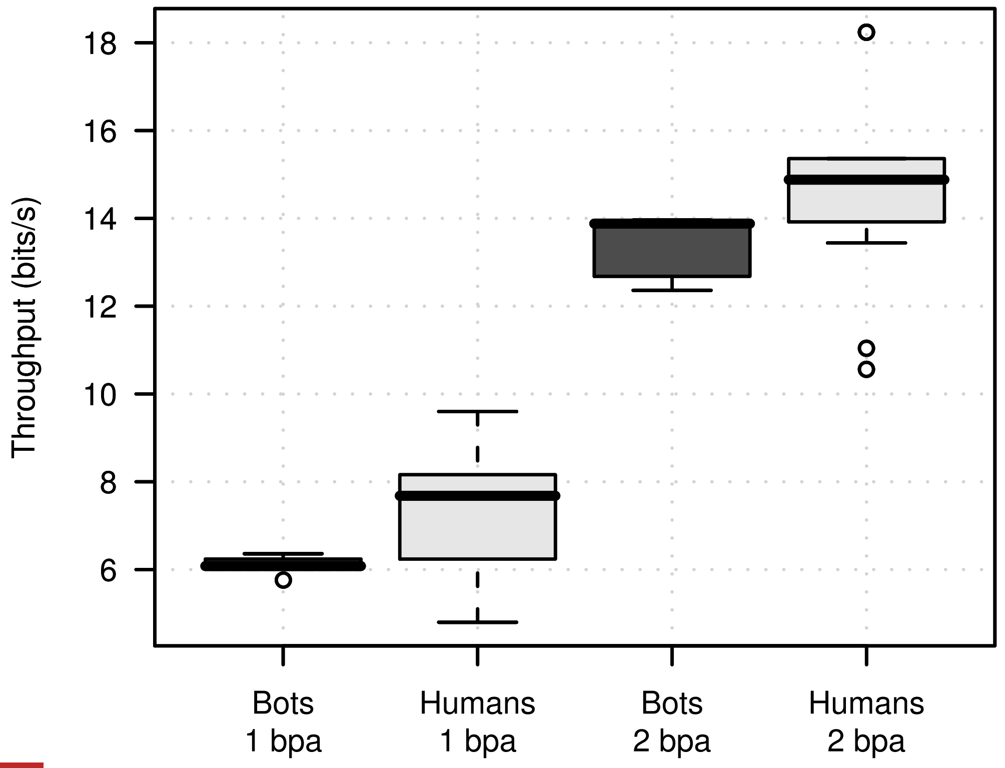
Throughput depending on packet loss

- 75 ms RTT and 0%, 0.5%, 1% loss (both directions)



Throughput human players vs. bots

- Games with 9 human players (25 ms RTT, 0% loss)



Ongoing and future work

- More trials to better understand performance and limitations
- Improve performance, especially for large RTTs
- Investigate similar covert channels for other games, immersive worlds

- Channel cannot be eliminated because player movement is intrinsic function of FPS games
- Blindly inserting noise does not work as covert sender can always send with higher ‘power’
- Develop efficient detection mechanism

Conclusions

- Developed novel covert channel in First Person Shooter (FPS) online game traffic
- Channel not limited to FPS games → other game types, immersive worlds
- Developed efficient mechanism for reliable transport
- Throughput up to 13–14 bits/s
  - Similar to other sophisticated covert channels
  - Sufficient for short text messages
- Covert channel is indirect and cannot be eliminated
- Detection is non-trivial (but probably possible)

Acknowledgements

- Many thanks to Lucas Parry, Lawrence Stewart, Warren Harrop, Amiel Hyde, Imrul Hassan, Mattia Rossi, Carl Javier, Tung Le, Lam Hoang Do and Kewin Stoeckigt for participating in the experiments