DISTRIBUTED SECURE CHANNEL

Daniel Smullen
Jonathan Gillett
Joseph Heron

BACKGROUND AND PROBLEM

Existing approaches use centralized infrastructure.

- TLS (used for HTTPS)
- Certificate Authorities

Other common methods use a single symmetric key for everyone.

- JGroups
- BitTorrent
- That's fine, but we can't tell you from someone else.
- No facility for issuing new keys, validating signatures to ensure integrity, providing authentication.

NOMENCLATURE

ECC	Elliptic Curve Cryptography	A very hard computational problem that allows cryptographers to generate keys that cannot be easily solved.
ECDSA	Elliptic Curve Digital Signature Algorithm	Used to digitally sign an entity.
Stream Cipher	Cryptography Term	Generates a stream of random data used to encrypt a message.
Grain128	Cryptographic Algorithm	A new, high performance and secure stream cipher.
IV	Initialization Vector	Used to ensure a message is never encrypted the same way twice.
CSPRNG	Cryptographically Secure Pseudorandom Number Generator	A secure random number generator for cryptographic applications.
ISAAC	Cryptographic Algorithm	A stream cipher used to generate a highly secure random number stream. Finding a pattern is almost impossible.
SHA-256	Hashing Algorithm	A secure 256-bit cryptographic hashing algorithm.
MD5	Hashing Algorithm	A secure 128-bit cryptographic hashing algorithm.
MAC	Message Authentication Code	Used to validate the integrity of a message to ensure it hasn't been altered.

CORE CONCEPTS I

Secure Channel

- Send messages securely over an insecure network so nobody can overhear you or tamper with messages (without someone knowing).

Public Key Verification

- Need to have a way of verifying someone's public key (to make sure it's them).
- Uses challenge/response authentication.

Public Key Signing

- Provide a way of guaranteeing the authenticity of a key.
- Just like a signature on paper – only you can sign it.

CORE CONCEPTS II

Authenticated Node Announcement

- Make sure every other node on the network knows when a new node's identity has been verified.

Joining the Network (Authentication)

- Requesting the encryption key from the available nodes.
- Now we can view messages on the network.

Stream Cipher

- What actually encrypts and decrypts the data?

Relay Chat

- We need an application for the DSC network. Why not IRC?

CRYPTOGRAPHIC COMPONENTS

Public Key Cryptography		Symmetric Key Cryptography (Stream Cipher)					
Public and Private Keys	Signing and Verification	Stream Cipher	Initialization Vectors	CSPRNG	Hash Functions		MAC
ECC	ECDSA	Grain128	Each node generates their own IV.	IVs are generated randomly using CSPRNG.	ISAAC	SHA-256	MD5	Each transaction signed using ECDSA.

AUTHENTICATION PROCEDURE

A user wants to join the DSC.

- JGroups cluster.

A group is selected to join.

Passphrase is entered.

- Their client signs their key;
- Their client sends out an authentication request to the cluster.

A previously authenticated client receives the request.

- They can approve or deny the request;
- They receive the signed public key.
- It is verified using the known authentication code.

If the key is valid, trust them.

- Or ignore them, if you don't trust them.

Each user that trusts the client announces the new relationship.

1:1 KEY EXCHANGE PROCEDURE

You've been authenticated, but you don't have your symmetric key yet.

Client sends a request for a key exchange.

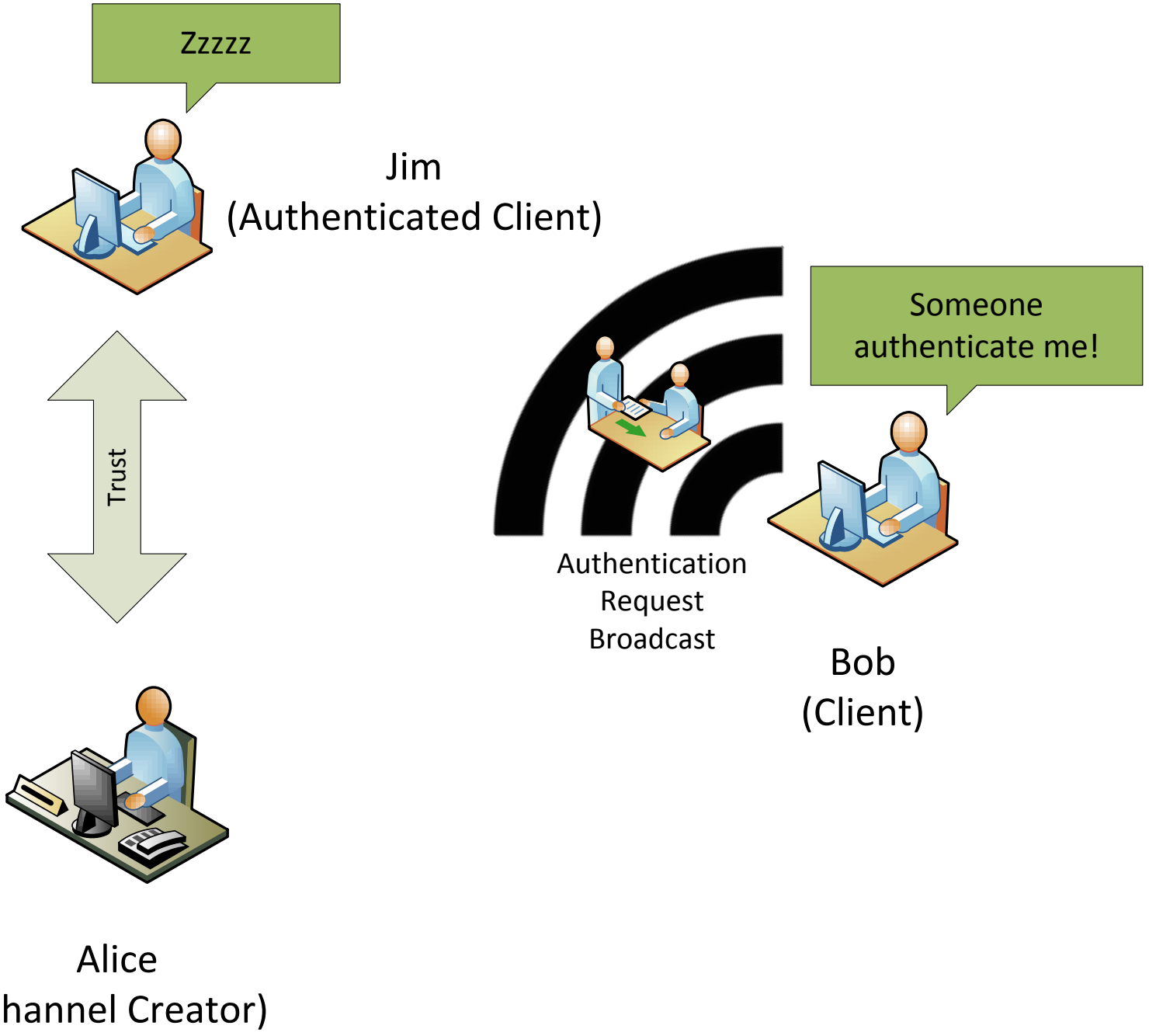
An already authenticated client checks your authenticity.

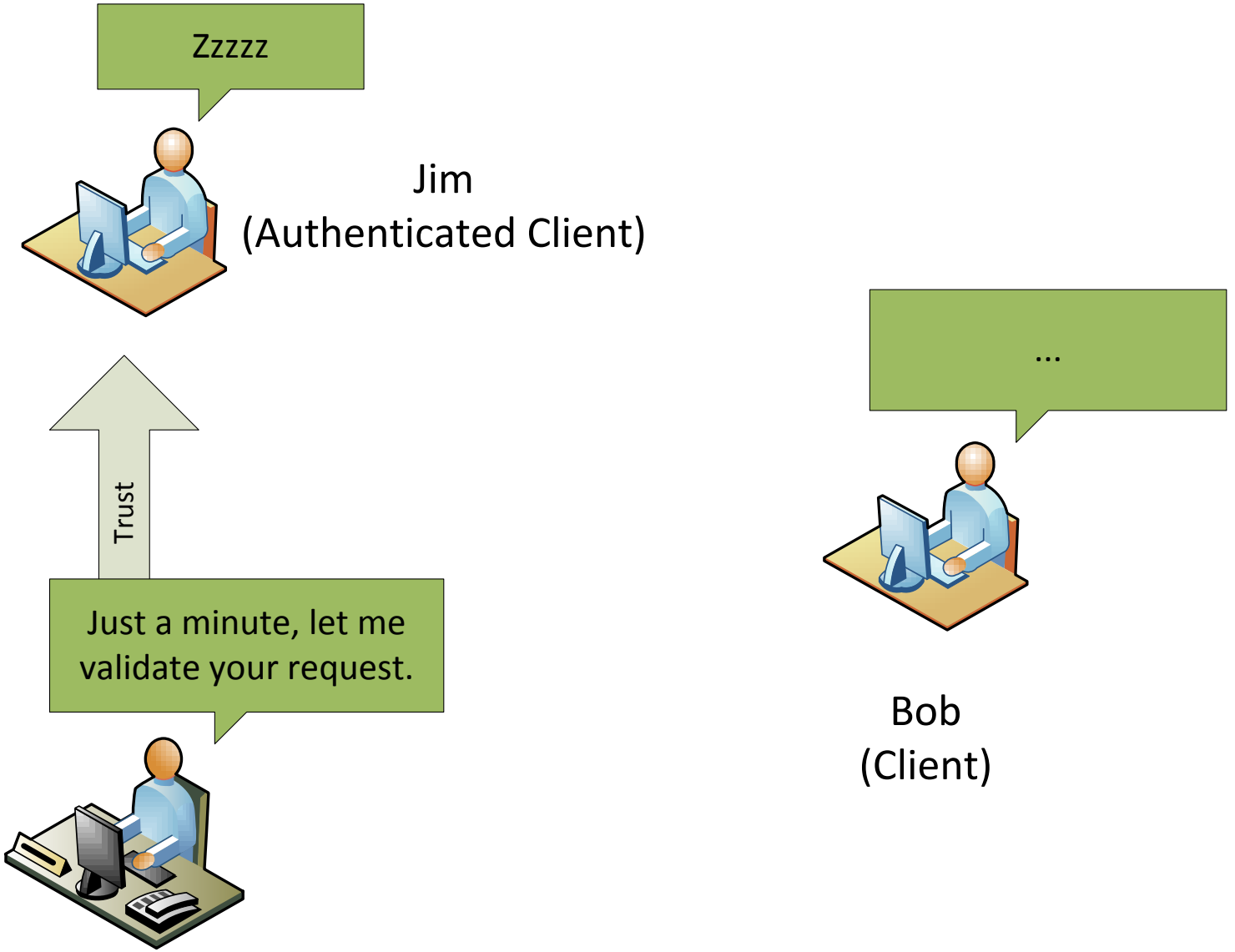
If you're authenticated, he'll encrypt the symmetric key (using your public key) and send it to you.

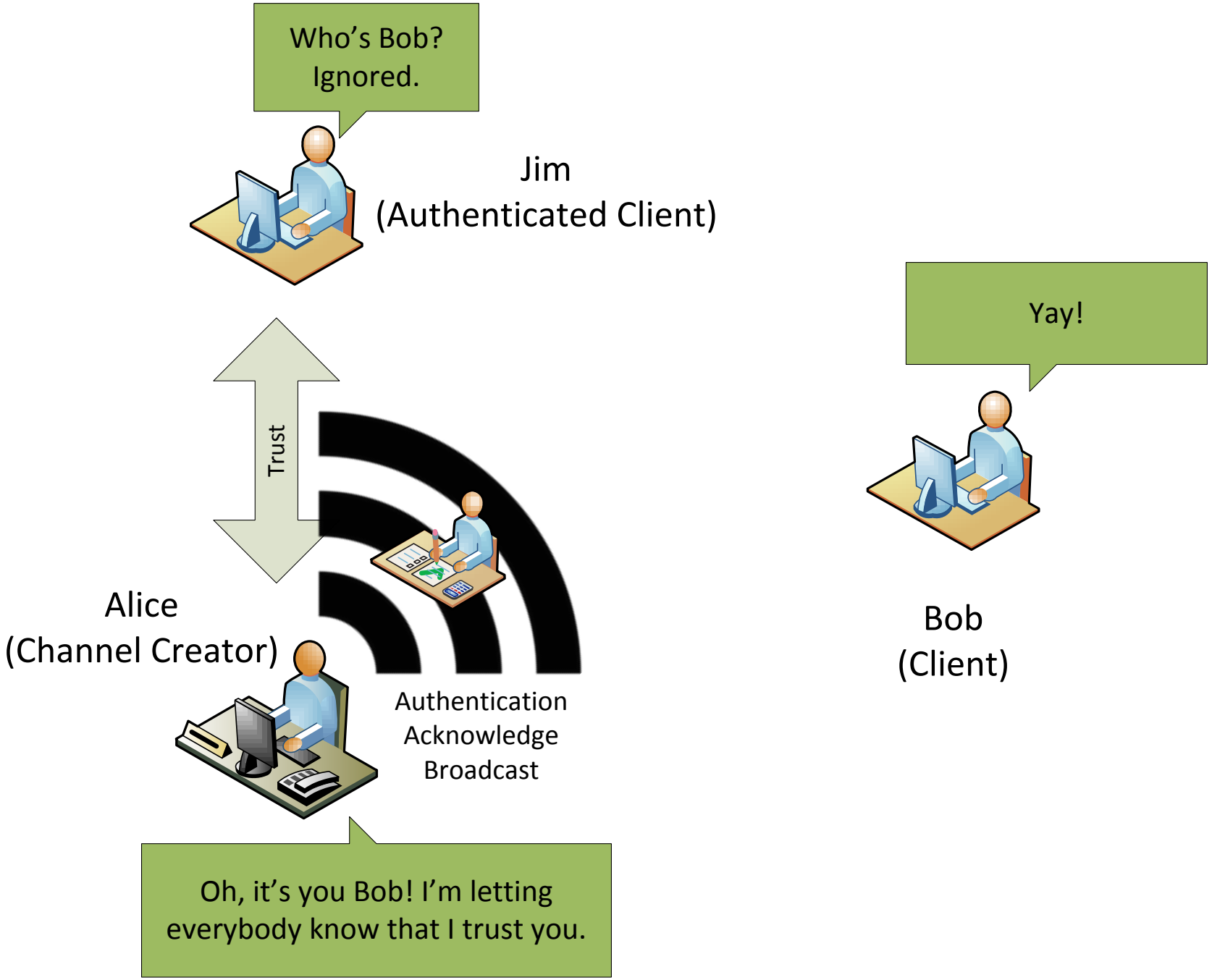
Now you have the symmetric key, and a trust relationship is created with that member.

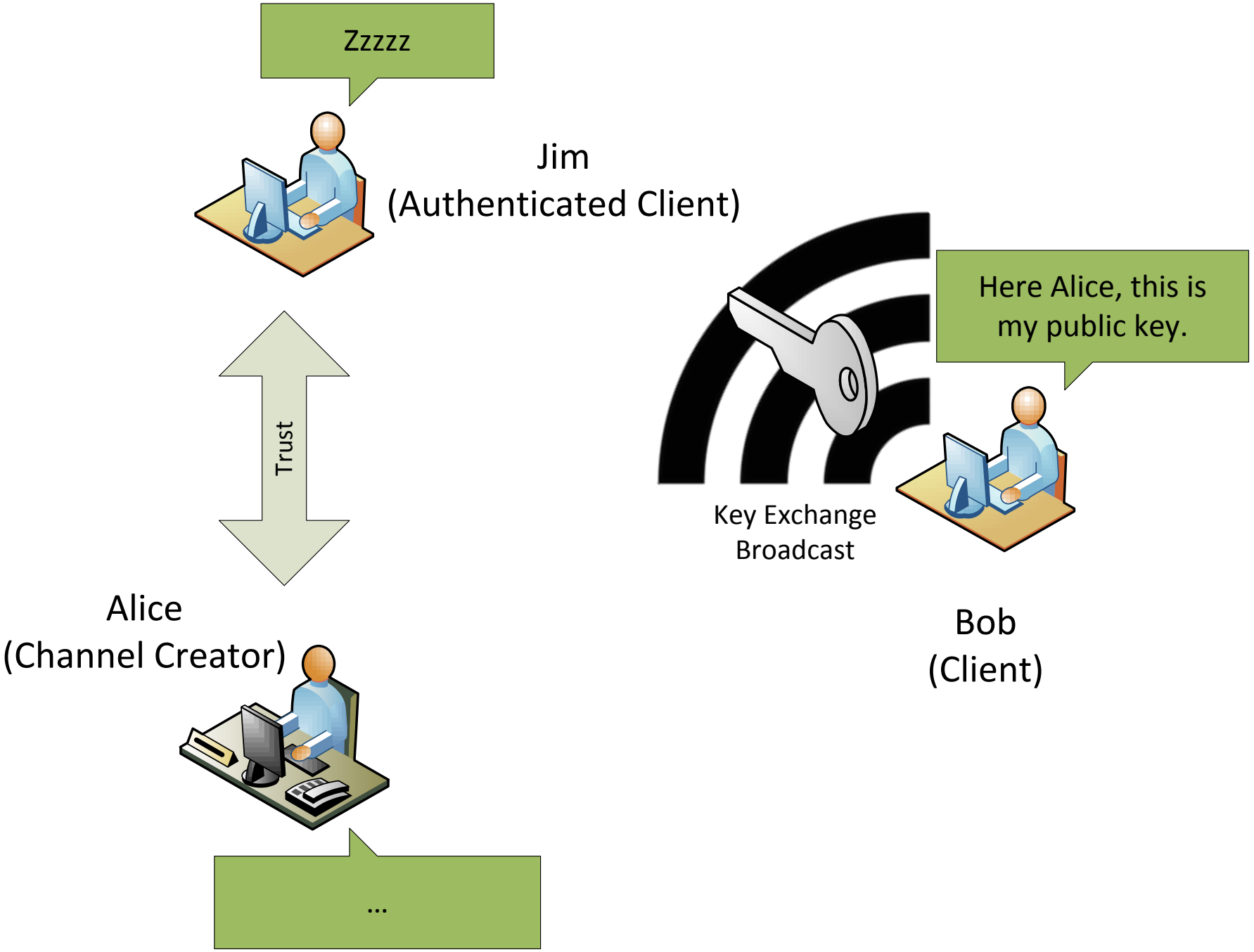
A SIMPLE KEY EXCHANGE

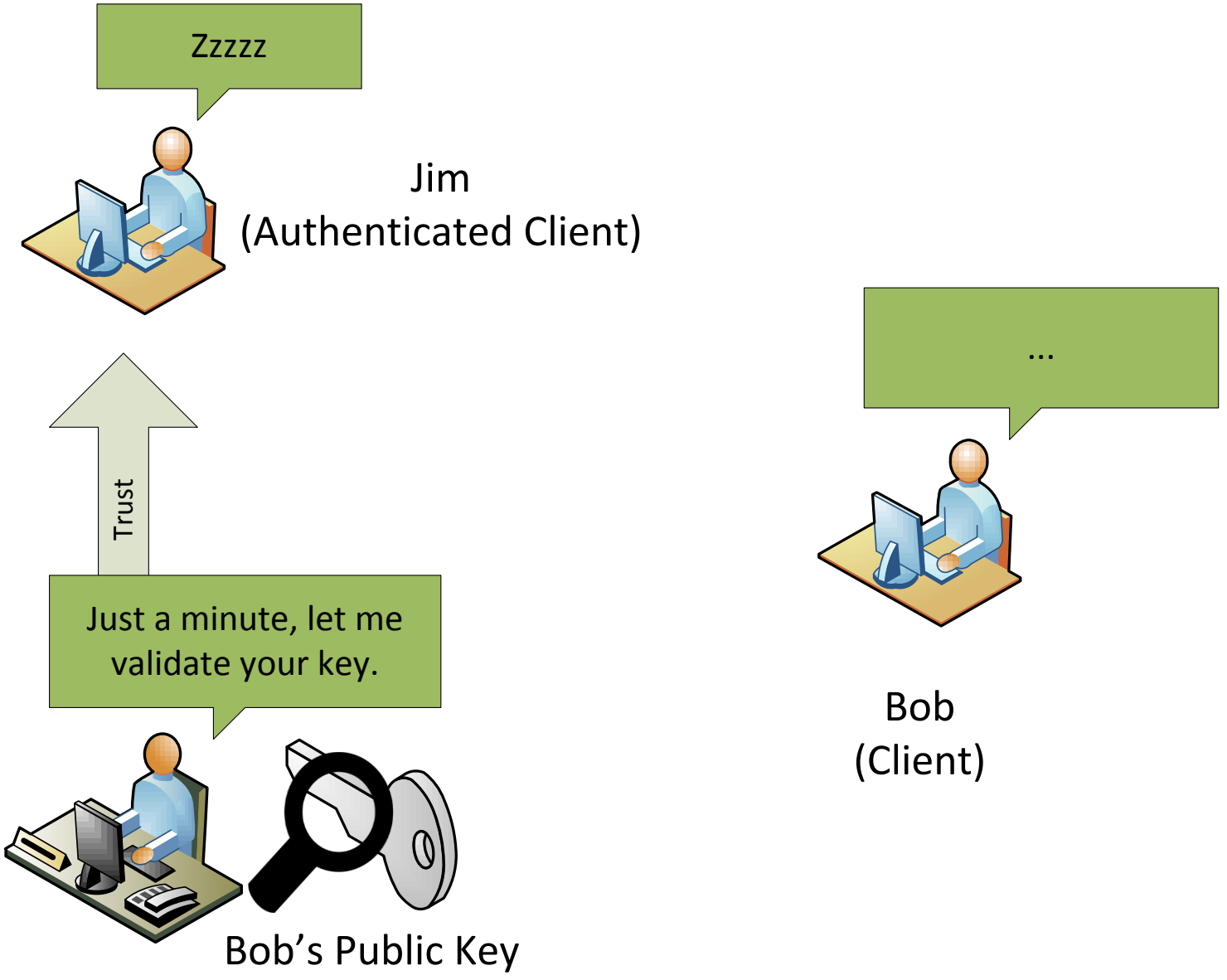
Starring: Alice and Bob
Featuring: Jim

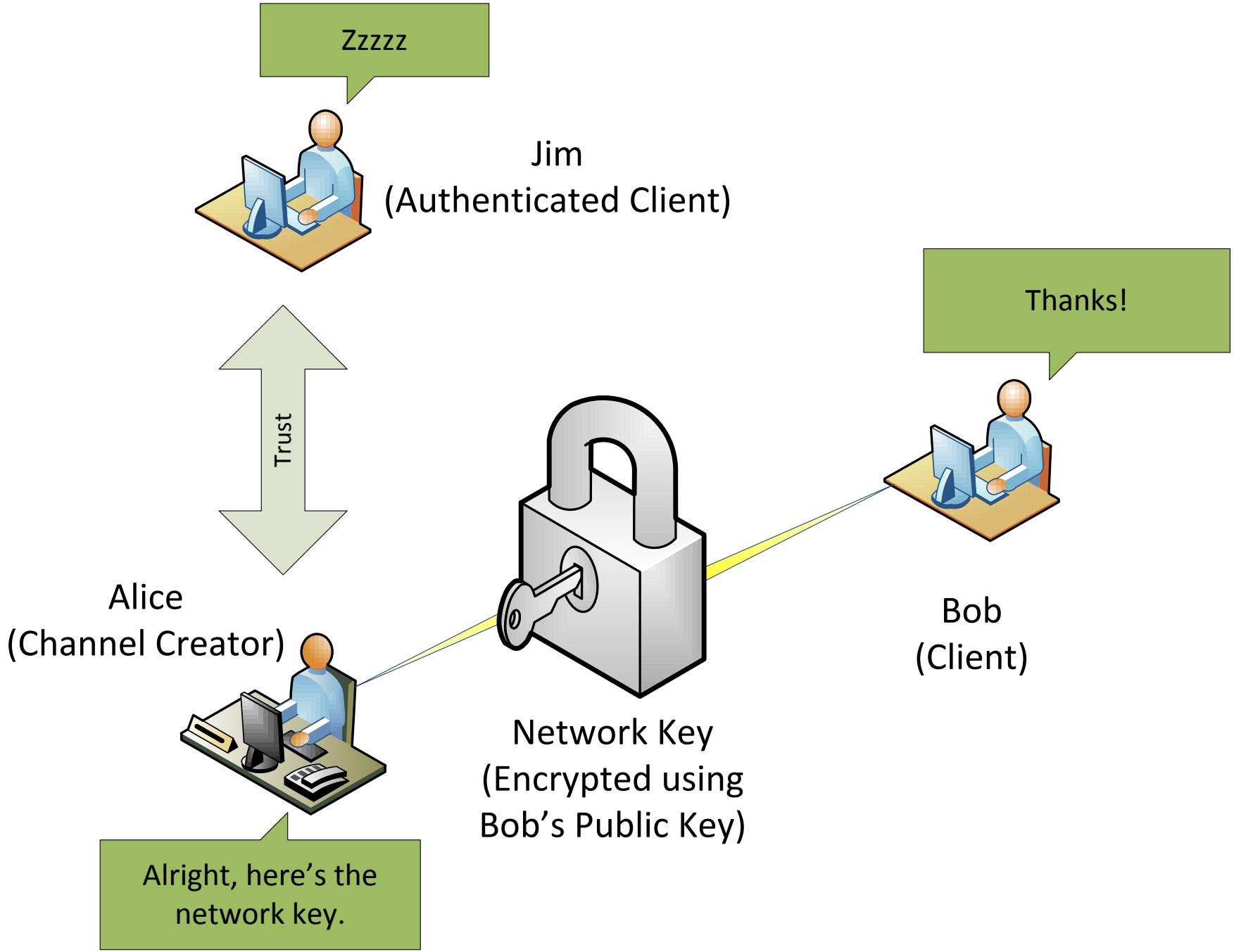

Zzzzz
Jim
(Authenticated Client)
Trust
Someone
authenticate me!
Authentication
Request
Broadcast
Bob
(Client)
Alice
(Channel Creator)

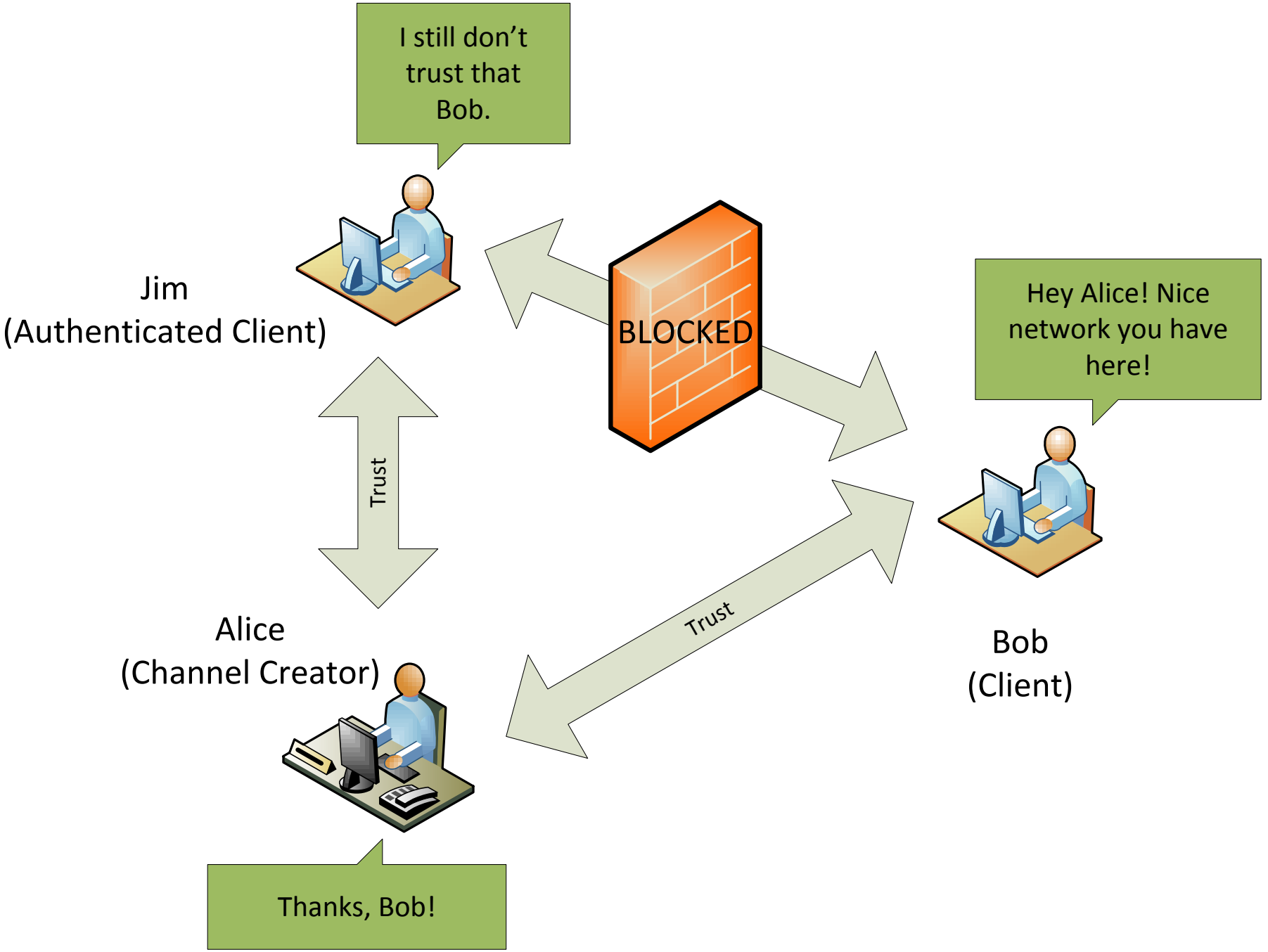

Zzzzz
Jim
(Authenticated Client)
...
Trust
Just a minute, let me validate your request.
Bob
(Client)
Alice
(Channel Creator)


Who's Bob?
Ignored.
Jim
(Authenticated Client)
Trust
Yay!
Alice
(Channel Creator)
Authentication
Acknowledge
Broadcast
Bob
(Client)
Oh, it's you Bob! I'm letting
everybody know that I trust you.


Zzzzz
Jim
(Authenticated Client)
Trust
Here Alice, this is my public key.
Key Exchange Broadcast
Alice
(Channel Creator)
Bob
(Client)
...


Zzzzz
Jim
(Authenticated Client)
...
Trust
Just a minute, let me validate your key.
Bob
(Client)
Bob's Public Key
Alice
(Channel Creator)


Zzzzz
Jim
(Authenticated Client)
Thanks!
Trust
Alice
(Channel Creator)
Bob
(Client)
Network Key
(Encrypted using
Bob's Public Key)
Alright, here's the
network key.


I still don't trust that Bob.
Jim
(Authenticated Client)
BLOCKED
Hey Alice! Nice network you have here!
Trust
Trust
Alice
(Channel Creator)
Bob
(Client)
Thanks, Bob!

HIGH LEVEL SYSTEM ARCHITECTURE

Message

Message Type Enumeration
- Authentication Request
- Authentication Acknowledge
- Key Exchange
- Key
- Message

Message Interface
- Defines the operations used for sending Message objects.

Operations

- Authentication Request
- Authentication Acknowledge
- Key Exchange
- Key
- Message

Internal Cryptography

- Public Key Cryptography
- Signing and Validation
- Stream Cipher
- Initialization Vector
- CSPRNG
- Hashing
- MAC

IRC Client

- Client classes for displaying text, nicknames.
- Authentication Requests
- Notification Prompts

Utility

Program State Class
- Globally synchronized state used to administrate system actions based on the system state.

Back Off Timer
- Used to prevent synchronization issues.

SYSTEM STATES

Untrusted

- Requesting Authentication
- Receiving Authentication Acknowledgement
- Sending Key Exchange Request
- Receiving Key Exchange

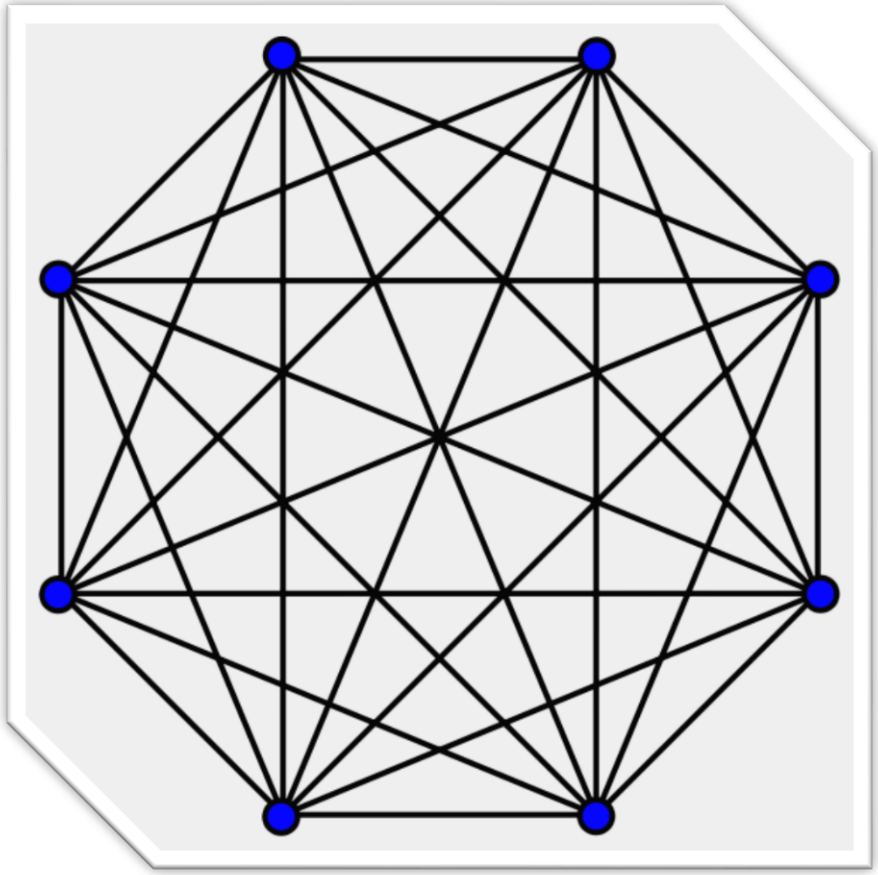
Trusted

- Receiving Authentication Request
- Authentication Decision
- Receiving Authentication Acknowledgement
- Sending Authentication Acknowledgement
- Awaiting Key Exchange Request
- Sending Key Exchange
- Receiving Message
- Sending Message

WHY USE THIS SYSTEM?

Why not just use public key cryptography for everything?

- How do I know the node providing me a public key is really the node it says it is?
  - Man in the middle attack vulnerability.
- If you have n nodes, the number of key exchanges that are required must form a complete graph of exchanges.
- The overhead for each client must be done n-1 times.
  - Huge computational overhead.



WHY USE THIS SYSTEM?

Why not just hard-code a symmetric key, like JGroups?

- This solution is insecure.
- No way of authenticating new nodes who want to join.
- Reprogramming/redistributing the key is problematic.
  - Cannot be deterministic.
  - Should be generated by CSPRNG.

```
34  # This is used for encrypting packets every network
35  # should have its own unique key.
36  network_key = 'password987access!'
37
38  # This is the name of the "hub" which is seen by the user
```

WHY USE THIS SYSTEM?

Why not just use WPA?

- WPA is designed for a single or multi-point access node that is centralized.
- WPA requires a fixed password.
- If you need to cycle the password, it's impossible to propagate to new nodes which use the old password.

INTERESTING EXTENSIONS/FUTURE WORK

A new cluster can be created at any time (even within a cluster), the initial nodes that create the cluster are the signing authority unless they add new signing authorities.

This cluster would have it's own unique symmetric key, and other *a priori* info required to join the network.

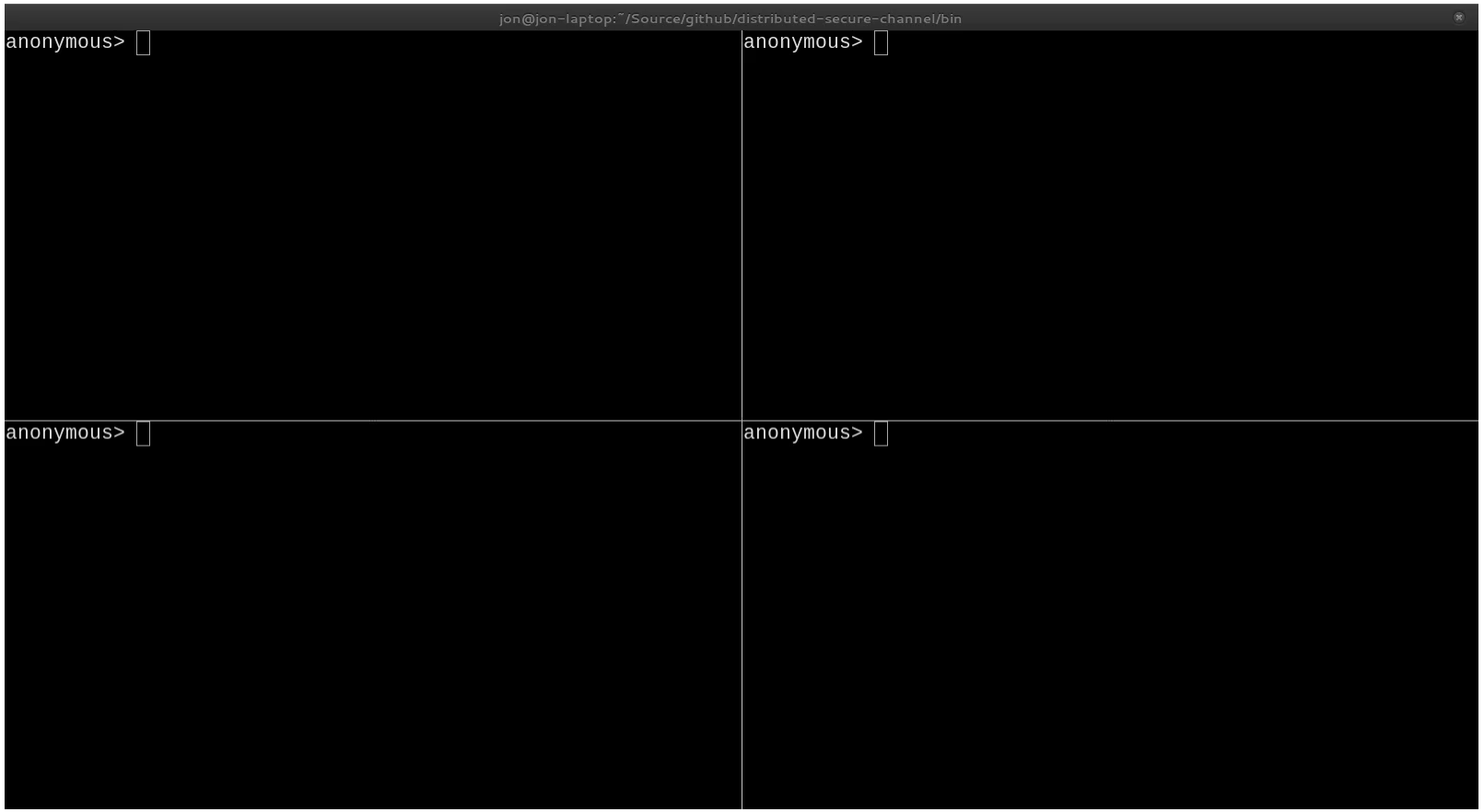
DEMO 1

Main Demonstration


jon@jon-laptop:~/Source/github/distributed-secure-channel/bin
anonymous>
anonymous>

DEMO 2

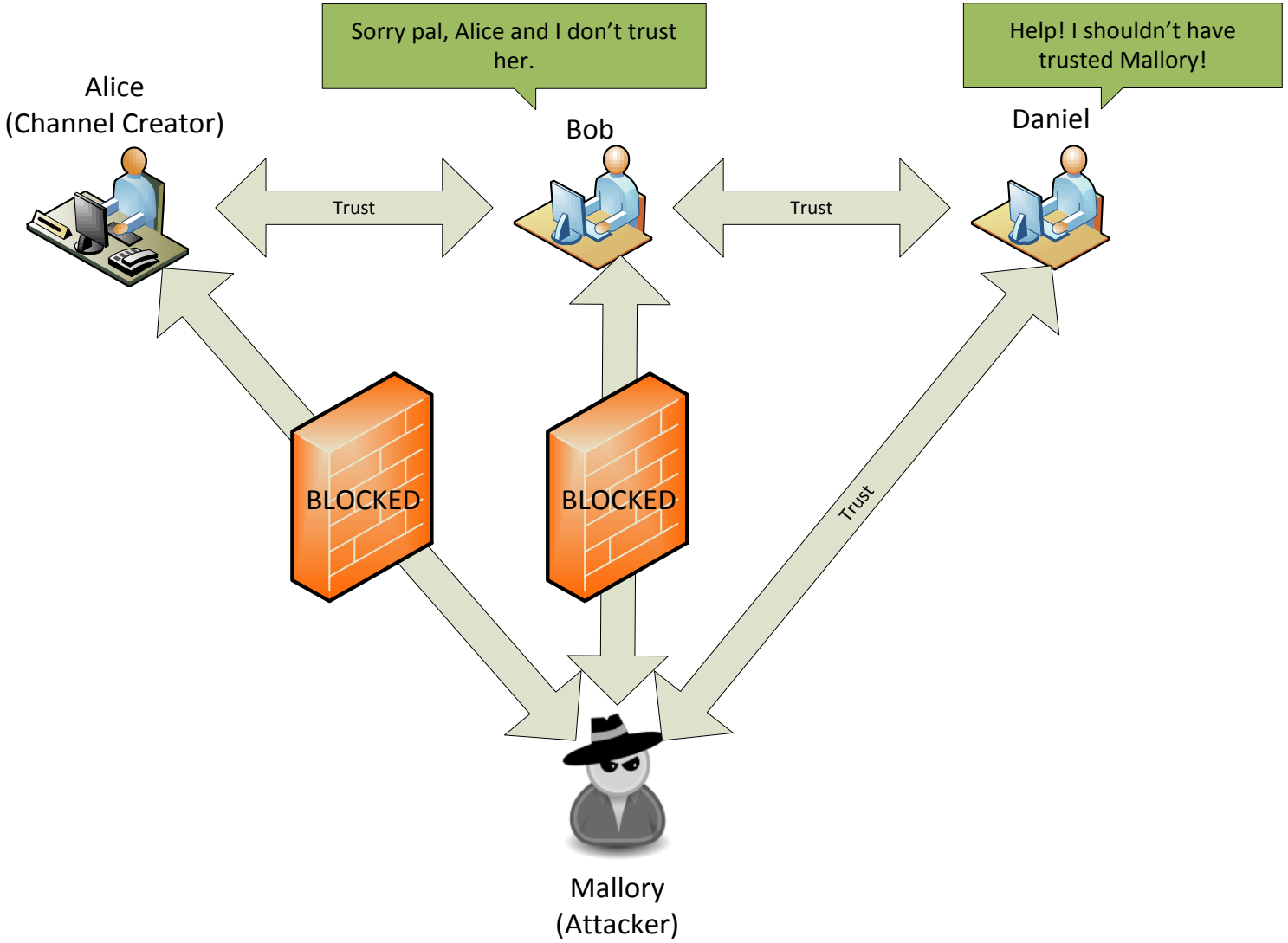
Denying an Attacker

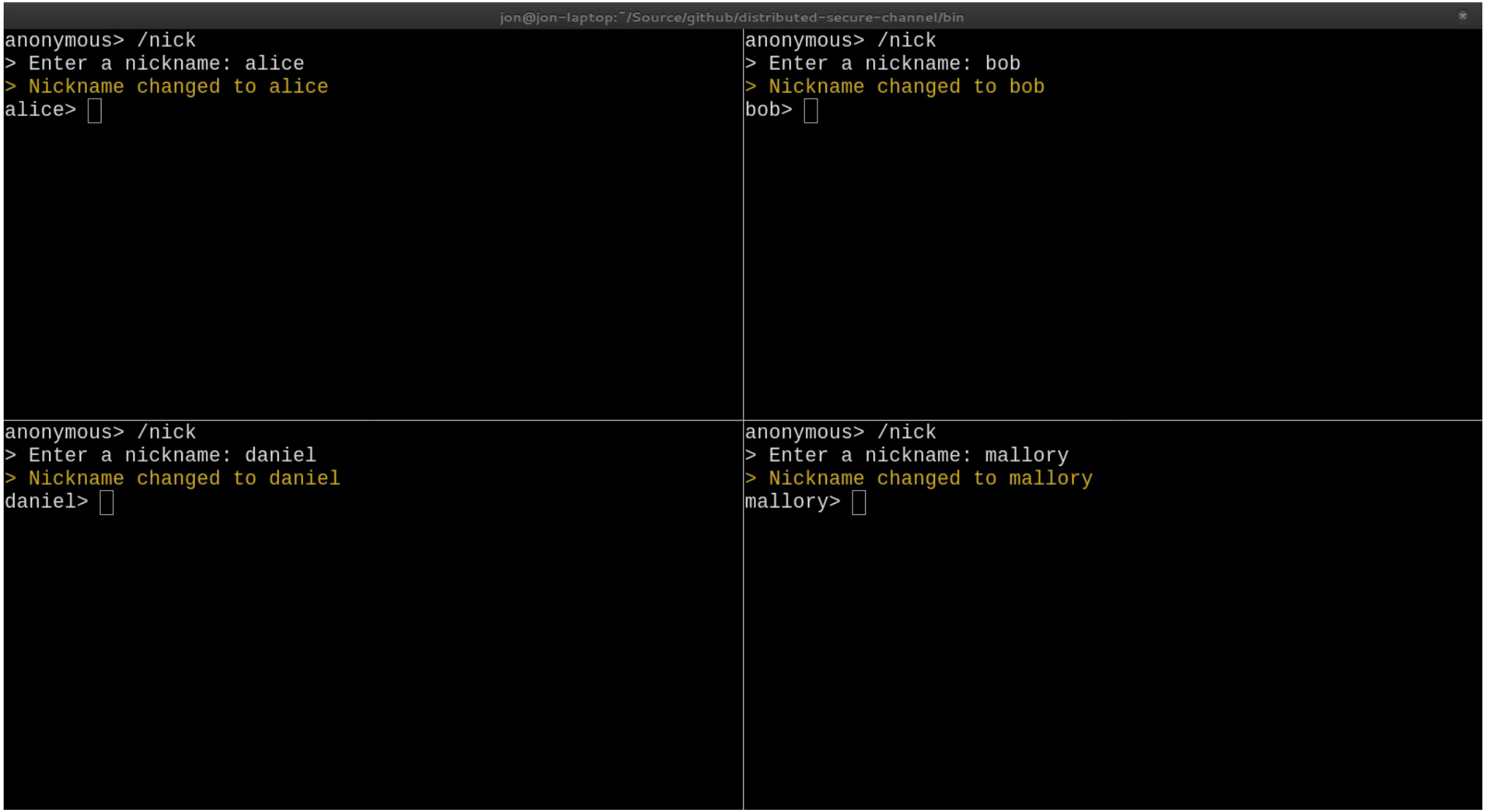

jon@jon-laptop:~/Source/github/distributed-secure-channel/bin
anonymous>
anonymous>
anonymous>
anonymous>

DEMO 3

Diminishing Access

DIMINISHING ACCESS




jon@jon-laptop:~/Source/github/distributed-secure-channel/bin
anonymous> /nick
> Enter a nickname: alice
> Nickname changed to alice
alice>
anonymous> /nick
> Enter a nickname: bob
> Nickname changed to bob
bob>
anonymous> /nick
> Enter a nickname: daniel
> Nickname changed to daniel
daniel>
anonymous> /nick
> Enter a nickname: mallory
> Nickname changed to mallory
mallory>