Encrypted PostgreSQL

PGCon 2009
Ottawa, Canada

Magnus Hagander
Redpill Linpro AB

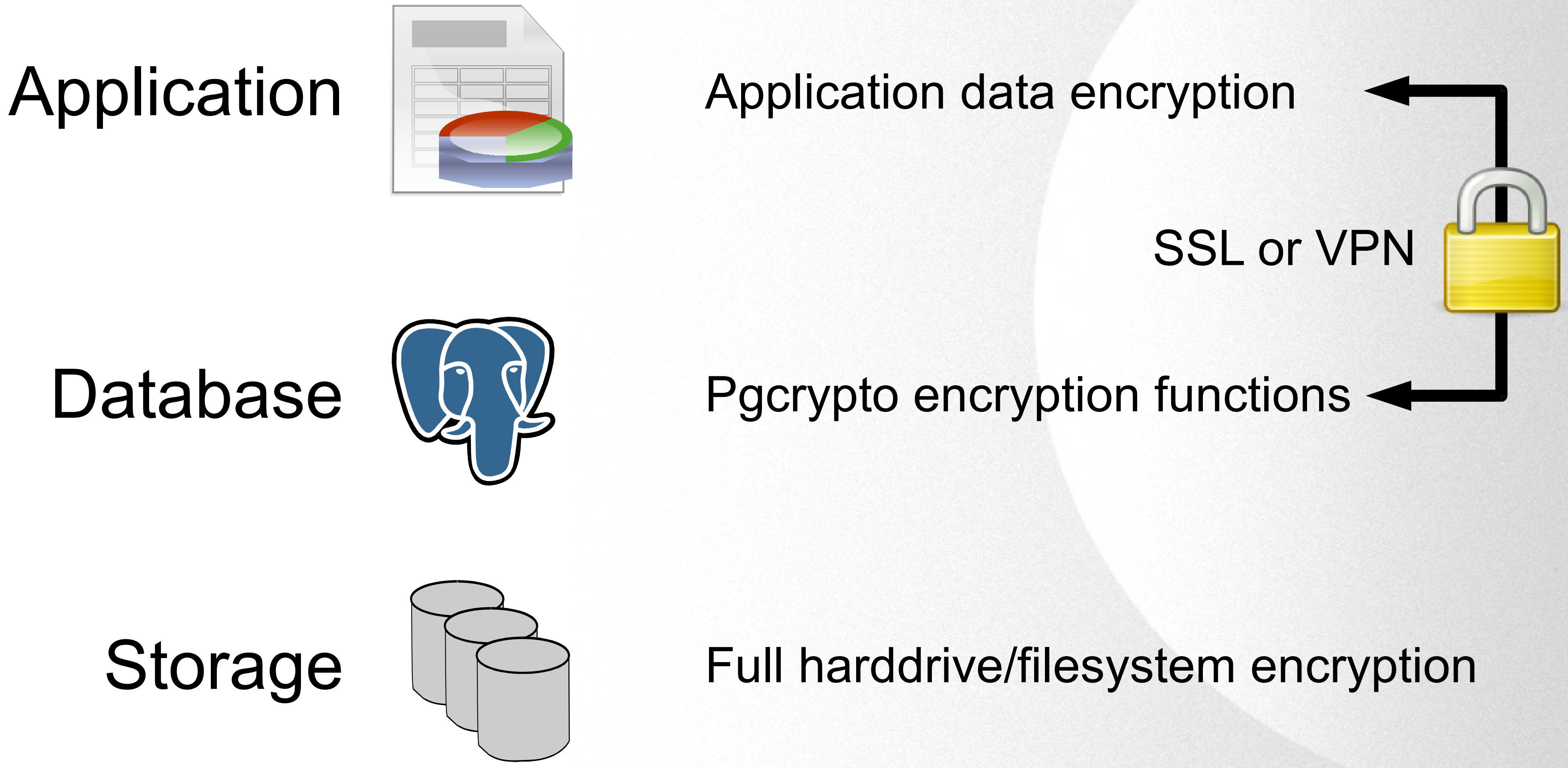
Decide what your threat is

- Everything comes at a cost
  - Performance or maintainability
- Encryption for the sake of encryption?
- Compliance/regulations?

Encryption at different layers

Application	Application data encryption
Database	Pgcrypto encryption functions
Storage	Full harddrive/filesystem encryption

Encryption at different layers


Application
Database
Storage
Application data encryption
SSL or VPN
Pgcrypto encryption functions
Full harddrive/filesystem encryption

Redpill Linpro

Application data encryption

- Independent of the database
- Implemented in the application layer
  - No, we won't talk about the myriad of options here

Harddrive/filesystem encryption

- Independent of the database
- Filesystem och block device level
- Needs to keep fsync behaviour!
- Keeps all database functionality
- Where to store the key?

Pgcrypto

- Encryption as database functions
- Client independent
- Don't forget to encrypt the connection!

Pgcrypto - challenges

- Encryption is easy
  - Relatively speaking
  - As long as you don't invent your own!
- Key management is not

Redpill Linpro

Pgcrypto – overview

- Raw encryption
- PGP compatible encryption
- Hashing

pgcrypto: raw encryption

```
SELECT encrypt(data, key, type)
SELECT decrypt(data, key, type)
SELECT encrypt_iv(data, key, iv, type)
```

- Type: bf-cbc, aes-cbc, ... (ecb supported, but..)
- Operates on bytea, returns bytea
- gen_random_bytes() can be used to create key

pgcrypto: PGP encryption

```
pgp_sym_encrypt(data, password[, opt])
pgp_sym_decrypt(data, password[, opt])
```

- Operates on text in plaintext, bytea in ciphertext
  - armor(), dearmor()
- Takes gpg style options like *ciper-algo=aes256*

Redpill Linpro

pgcrypto: PGP encryption

```
pgp_sym_encrypt(data, password[, opt])
pgp_sym_decrypt(data, password[, opt])
```

- Public key encryption also supported, but no key generation
- Will detect wrong key/corrupt data

pgcrypto: Hashing

- `SELECT digest(txt, type)`
  - Returns bytea, use encode() to get hex
  - Md5, sha1, sha<more>

-
  ```
  SELECT encode(
     digest('lolcats!', 'sha256'),
     'base64')
  ```

pgcrypto: Hashing

- `SELECT crypt('secret', gen_salt('bf'))`
  - Stores salt as part of hash
  - Autodetects algorithm
  - md5, bf, etc
- `SELECT hash=crypt('secret', hash)`

Key management

- Where to store the key
- How to protect the key
- How to access the key
- How to do key recovery

Searching encrypted data

- Sorry, can't really be done by index
- Match encrypted data for raw encrypted *without* padding
  - But this decreases security
  - And does «is equal» matching only
- Index on expression
  - But why did you encrypt in the first place?

SSL

SSL secured connections

- Encryption
- Man-in-the-middle protection
- Authentication

SSL secured connections

- Enabled on the server (ssl=yes)
- Optionally required through pg_hba
- Optionally required in libpq

Redpill Linpro

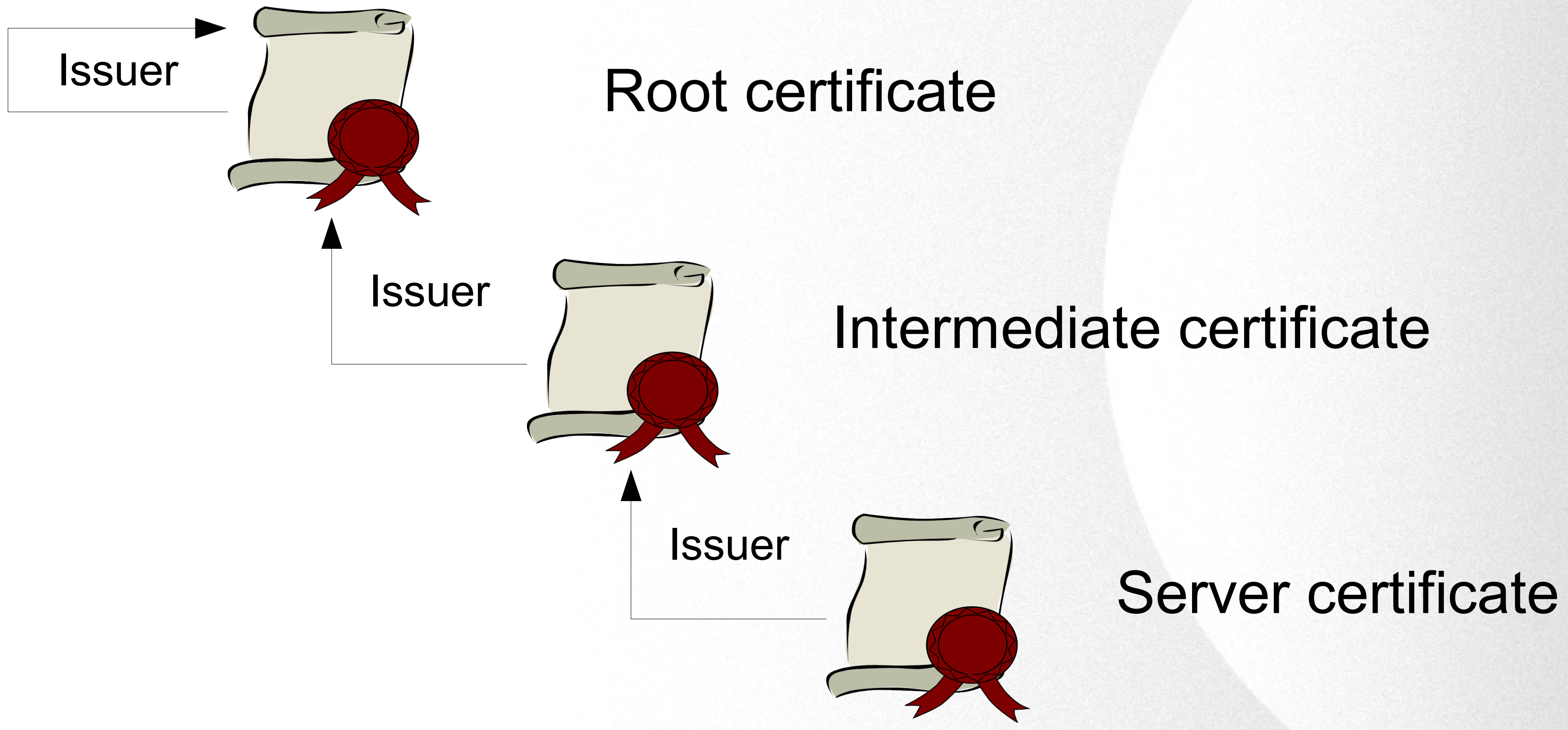
SSL secured connections

- Need to protect data in *both* directions
- For example username/password
- Must *know* before connection is started
  - Unknown equals unprotected

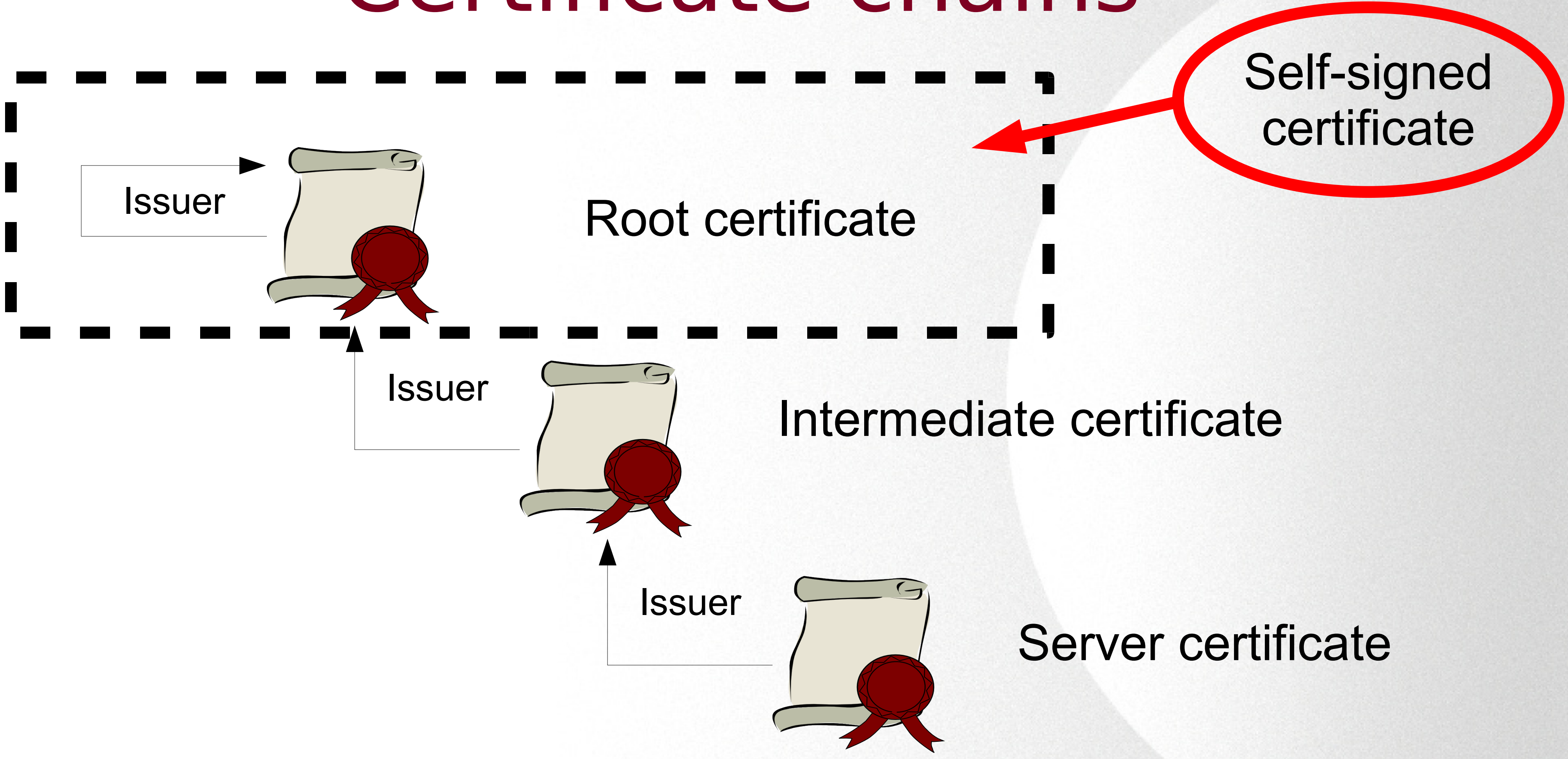
SSL encryption

- SSL *always* requires a server certificate
- Can be self-signed
- Does not need to be known by client

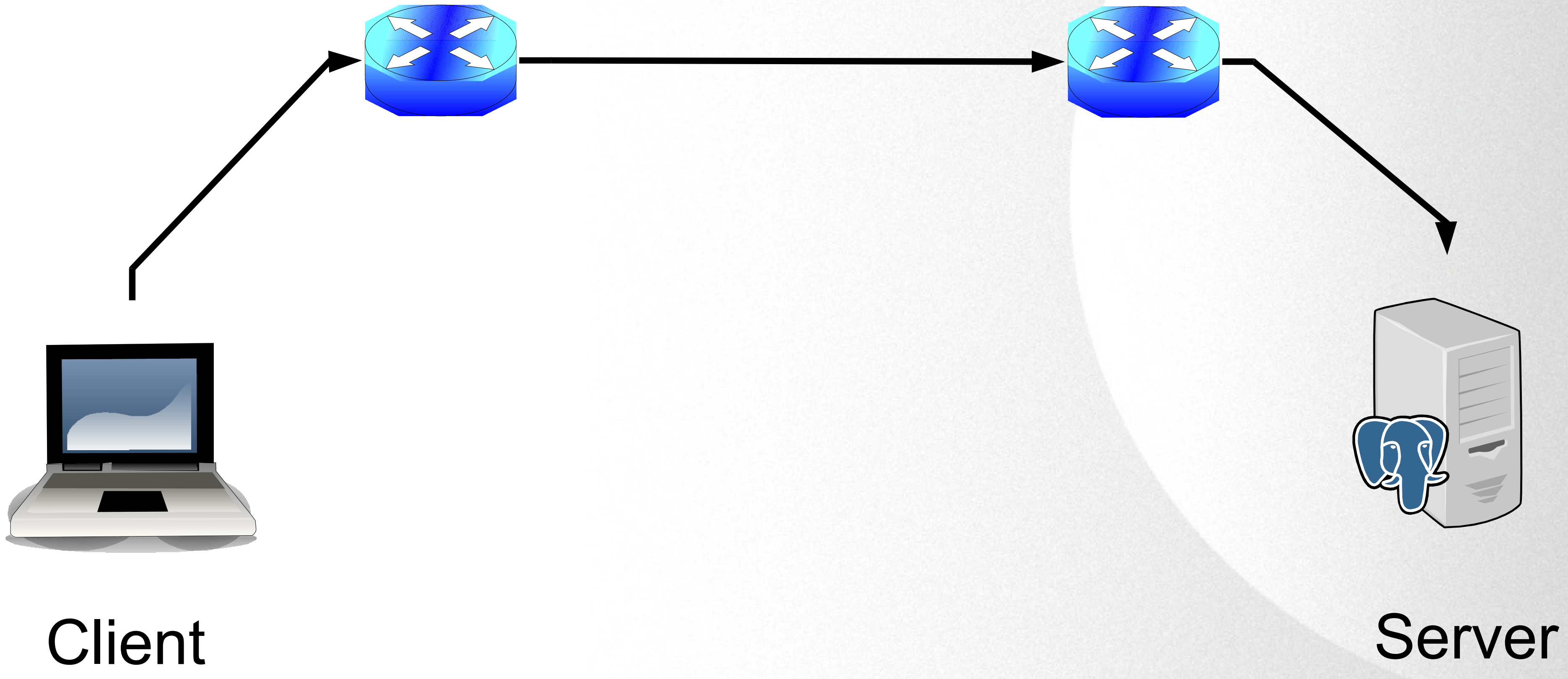
Certificate chains



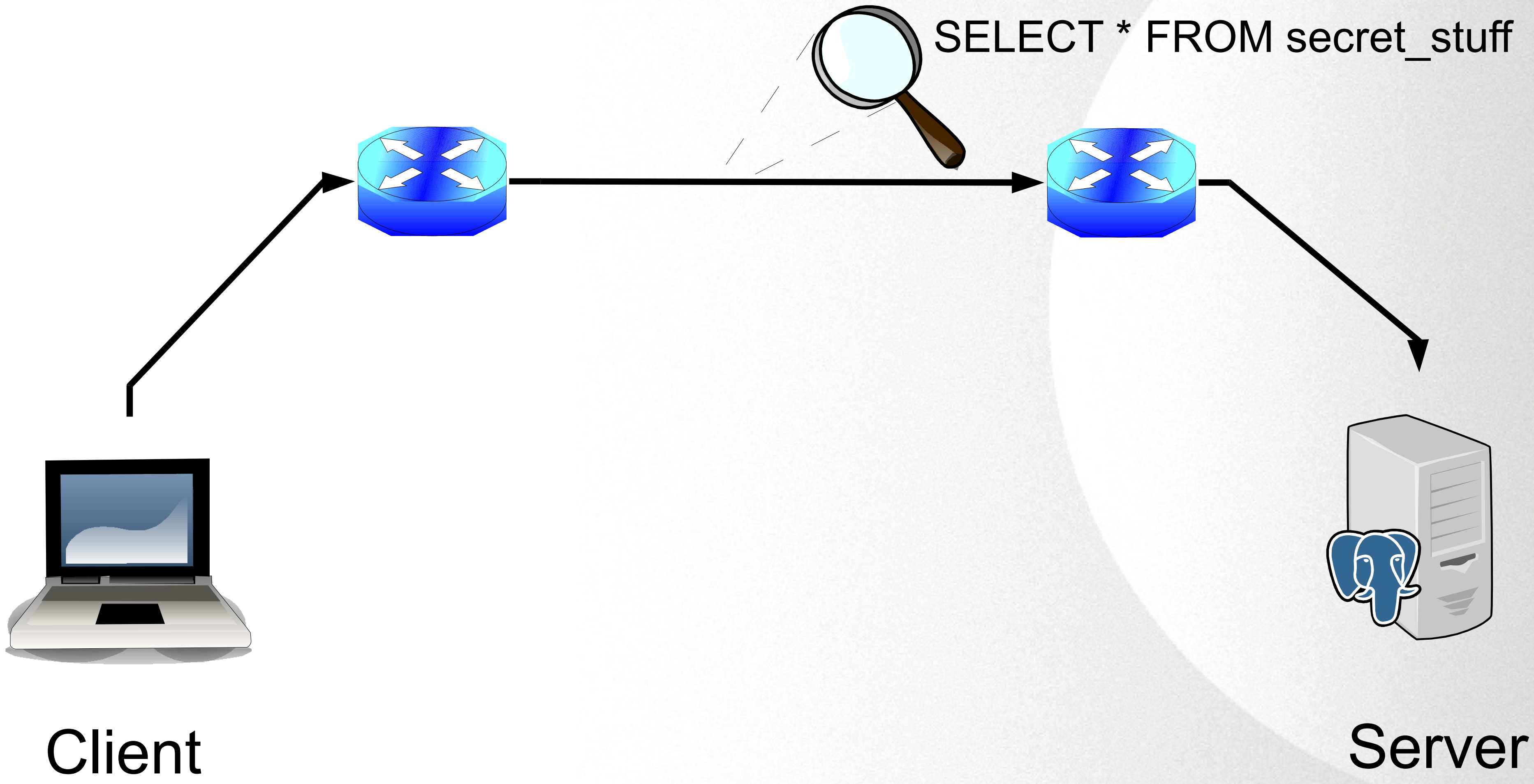
Certificate chains



SSL secured connections



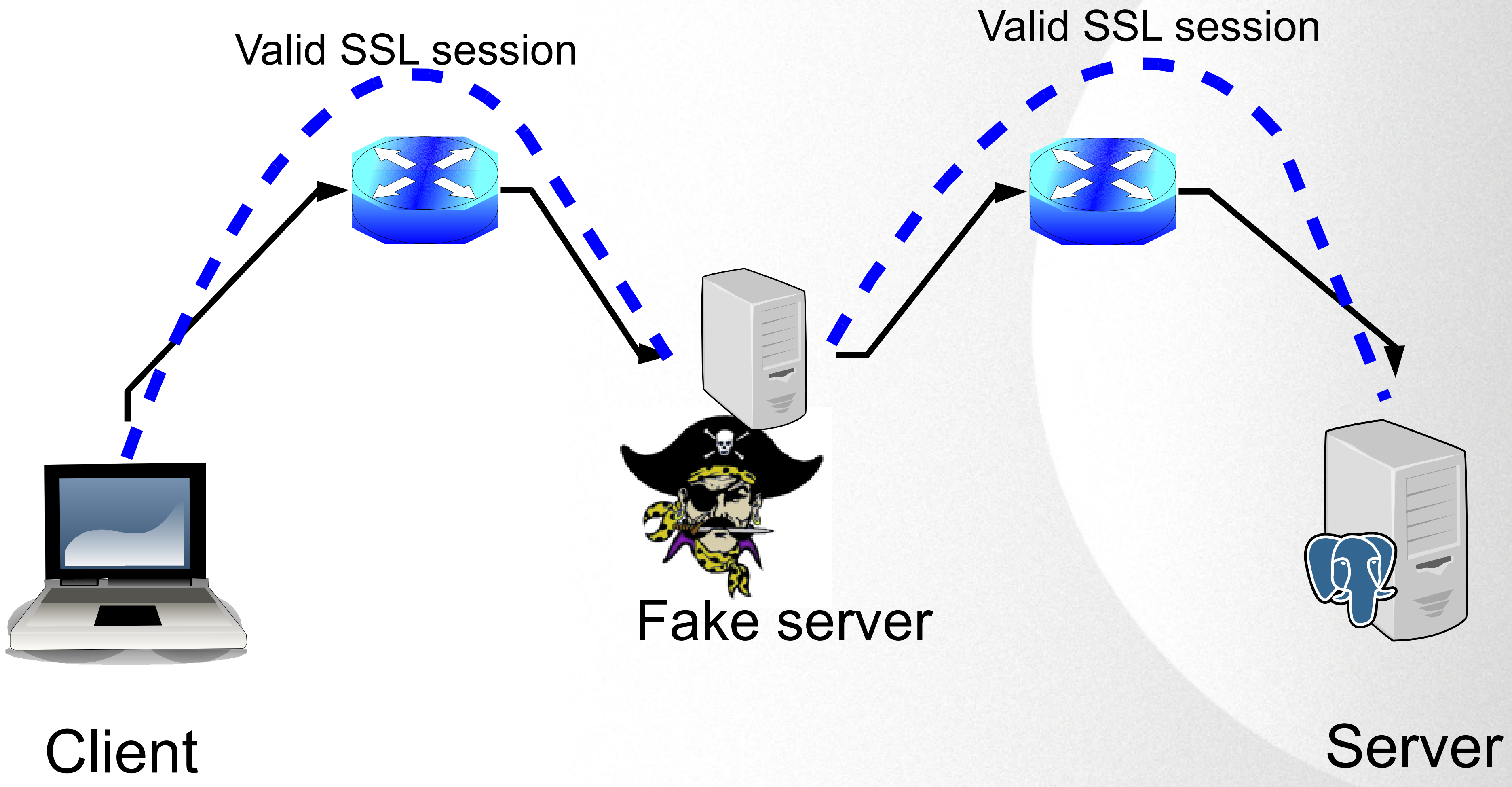
Threats handled by SSL: Eavesdropping



Eavesdropping

- Prevented by encrypting all data
- Key negotiation is automatic
- Server certificate used but not verified

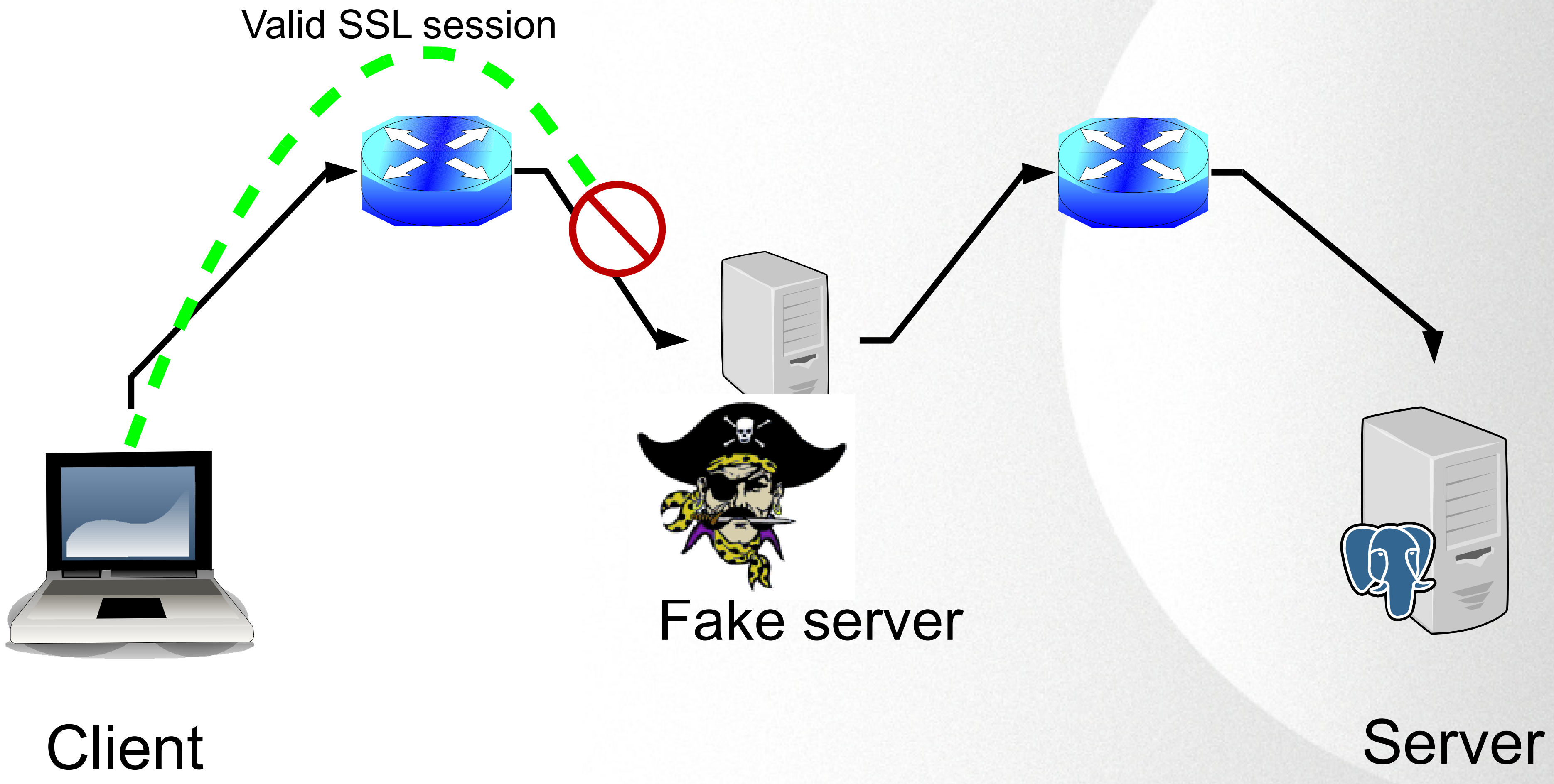
Threats handled by SSL: Man in the middle



SSL server verification

- On top of encryption
- Validate that the server is who it claims to be
- CA issues certificate, can be self-signed
- CA certificate known by client

Threats handled by SSL: Man in the middle



SSL client authentication

- On top of encryption
- Normally on top of server verificateion, but not necessary
- CA issued certificate on *client*
- Match *CN* on certificate to user id
- Protect client certificate!

SSL in libpq

- Controlled by *sslmode* parameter
- Or environment *PGSSLMODE*
- For security, must be set on client
  - Remember, *unknown = unsecure*

Redpill Linpro

Summary of libpq SSL modes

	Protect against		Compatible with server set to...		Performance
Client Mode	Eavesdrop	MITM	SSL required	SSL disabled	overhead
disable	no	no	FAIL	works	no
allow	no	no	works	works	If necessary
prefer	no	no	works	works	If possible
require	yes	no	works	FAIL	yes
verify-ca	yes	yes	works	FAIL	yes
verify-full	yes	yes	works	FAIL	yes

Summary of libpq SSL modes

	Protect against		**Compatible with server set to...**		**Performance**
Client Mode	**Eavesdrop**	**MITM**	**SSL required**	**SSL disabled**	**overhead**
disable	no	no	FAIL	works	no
allow	no	no	works	works	If necessary
prefer	no	no	works	works	If possible
require	yes	no	works	FAIL	yes
verify-ca	yes	yes	works	FAIL	yes
verify-full	yes	yes	works	FAIL	yes

Redpill Linpro

Summary of libpq SSL modes

	Protect against		Compatible with server set to...		Performance
Client Mode	**Eavesdrop**	**MITM**	**SSL required**	**SSL disabled**	**overhead**
disable	no	no	FAIL	works	no
allow	no	no	works	works	If necessary
prefer	no	no	works	works	If possible
require	yes	no	works	FAIL	yes
verify-ca	yes	yes	works	FAIL	yes
verify-full	yes	yes	works	FAIL	yes

Summary of libpq SSL modes

	Protect against		Compatible with server set to...		Performance
Client Mode	Eavesdrop	MITM	SSL required	SSL disabled	overhead
disable	no	no	FAIL	works	no
allow	no	no	works	works	If necessary
prefer	no	no	works	works	**If possible**
require	yes	no	works	FAIL	yes
verify-ca	yes	yes	works	FAIL	yes
verify-full	yes	yes	works	FAIL	yes

Summary

- Only encrypt what you really need
- Only encrypted *where* you really need
- Key management is *hard*
- Many use-cases are very narrow

Redpill Linpro

Encrypted PostgreSQL

Questions?

magnus@hagander.net
http://blog.hagander.net