A Comparison of PostgreSQL Encryption Options

Syed Faisal Akber, Staff Technical Support Engineer

Dong Ye, Staff Engineer

Agenda

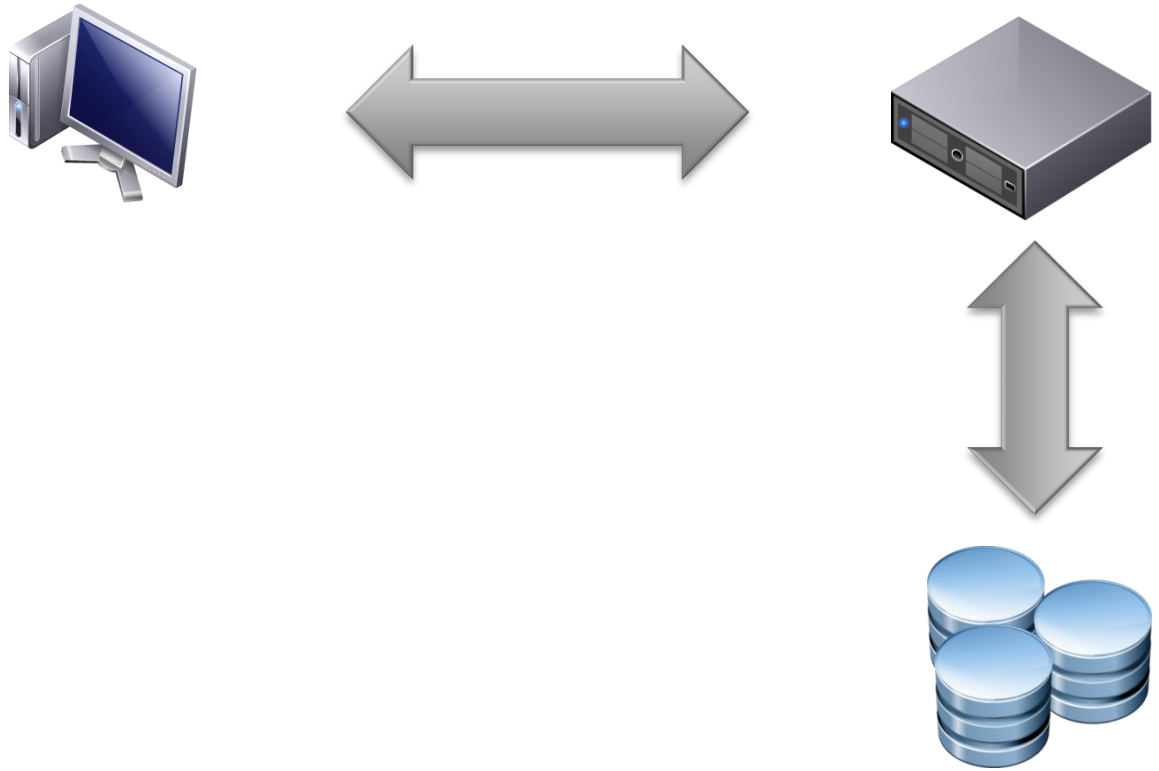
- Why encryption?
- Some Postgres encryption options
- Performance results
- Real-world use cases
- Conclusions

Why Encrypt Data?

- Protect sensitive information
- Prevent identity theft
- Satisfy paranoia

- Comply with laws and standards (SOX, HIPPA, PCI, …)

Typical Architecture



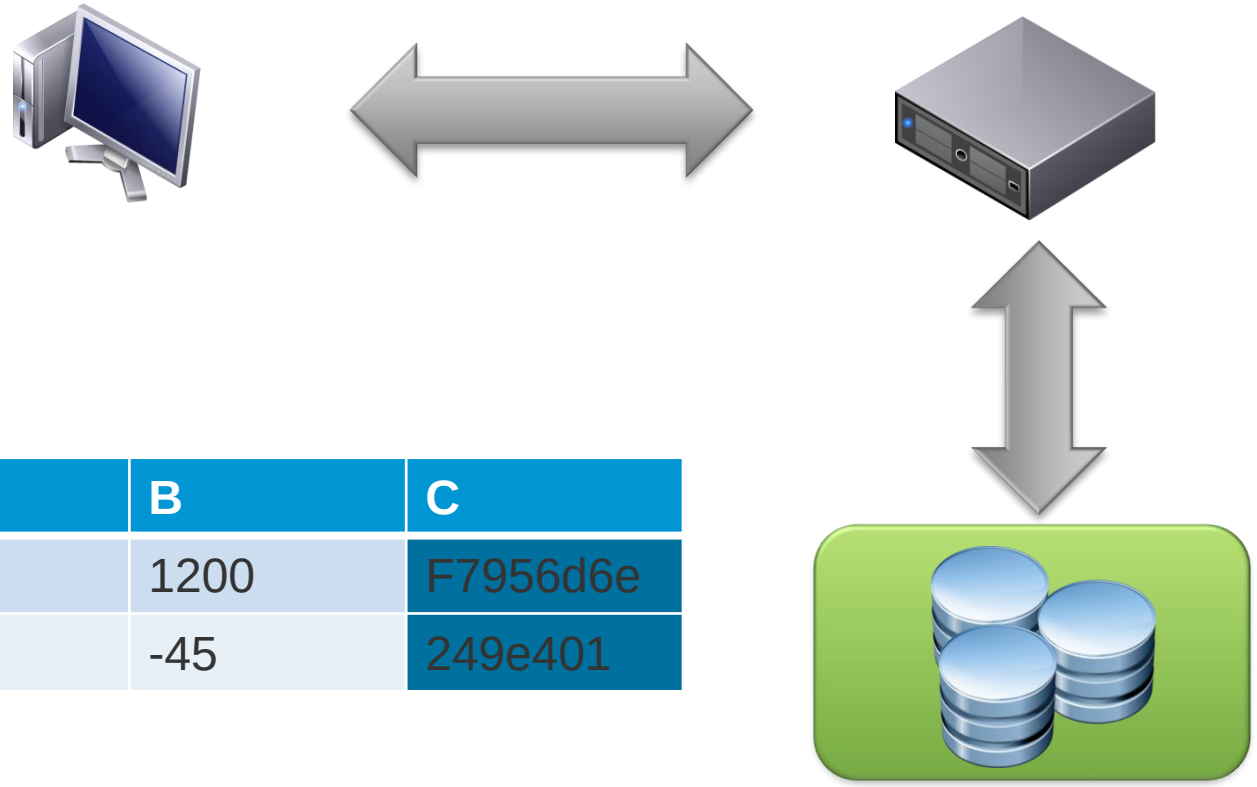
Postgres Encryption Options

- **Where?**
  - Encrypting Specific Columns
  - Encrypting Data Partitions
  - Encrypting Data Across Network
- **Who?**
  - Database Server/Client Communication over SSL
  - Complete Application Encryption

Encrypting Specific Columns

- **Why?**
  - Offload
  - Centralize
- **Use the pgcrypto module**
- **Require application change**

Encrypting Specific Columns: Diagram



A	B	C
1	1200	F7956d6e
2	-45	249e401

Specific columns are protected

Encrypting Specific Columns: pgcrypto

- **Provide a number of functions**
  - General hashing functions
  - Password hashing functions
  - PGP functions
  - RAW encryption/decryption functions

Using pgcrypto

- **Build Server and Extension, Use Extension**

```
./configure --with-openssl
make
make install
cd contrib/pgcrypto
make
make install
```

```
pgbench=# CREATE EXTENSION pgcrypto;
```

- **Augment DML in application**

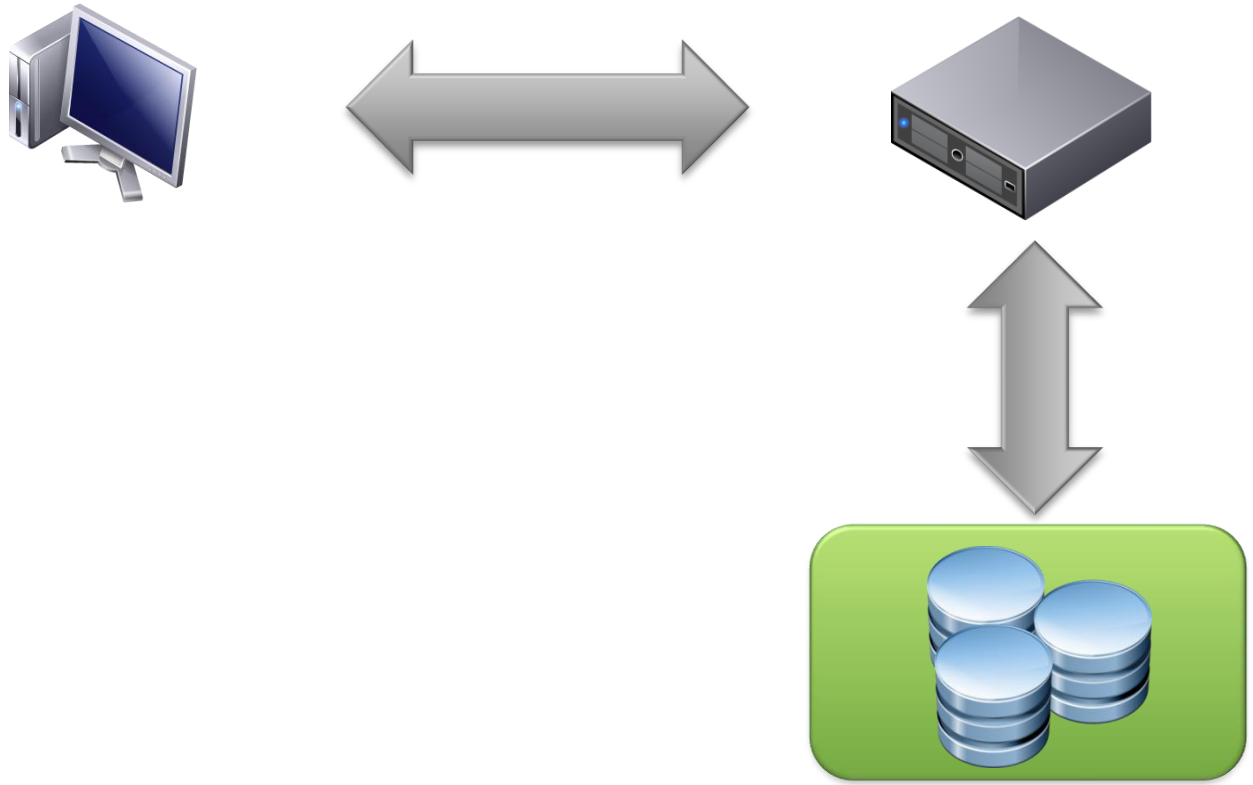
INSERT Example

```
INSERT INTO z (a, b, c) VALUES (3, 34500, encrypt('Test'::bytea,
'key'::bytea, 'aes'));
```

SELECT Example

```
SELECT a, b, convert_from(decrypt(c, 'key'::bytea, 'aes'),
current_setting('server_encoding'))::int AS c FROM z WHERE a = 1;
```

Encrypting Data Partition: Diagram



Encrypting Data Partition (Filesystem)

- **Prepare an encrypted filesystem with dm-crypt**

```
dd if=/dev/zero of=/data/crypt count=8 bs=1G
chmod 600 /data/crypt
losetup /dev/loop0 /data/crypt
cryptsetup -y create secretfs /dev/loop0
cryptsetup status secretfs
mke2fs -j -O dir_index /dev/mapper/secretfs
tune2fs -l /dev/mapper/secretfs
mkdir /mnt/secretfs
mount /dev/mapper/secretfs /mnt/secretfs/
```

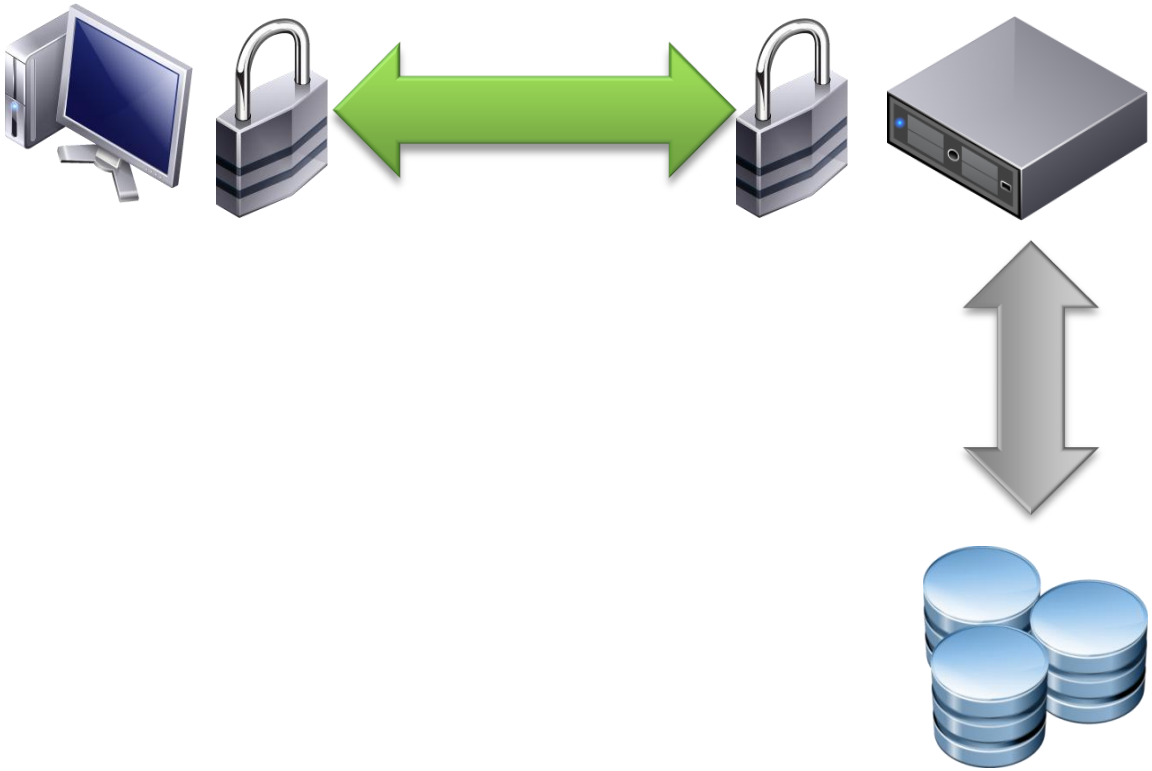
- **Run initdb on the encrypted filesystem**
- **Start Postgres server**

Encrypting Data Across Network

Two main methods

- Postgres built-in SSL
- SSH tunnel

Encrypting Data Across Network



Encrypting Data Across Network: SSL

- Facility exists in Postgres
- Configure server
- Configure SSL flag in client
- May need to open ports in firewall/router

Cisco PAT configuration in Cisco IOS

```
ip nat inside source static tcp 10.4.3.2 5432 interface Serial0 5432
```

Server Configuration

Build Server

```
./configure --with-openssl
make
make install
```

Create SSL Keys and Sign Certificate

```
openssl req -new -text -out server.req

openssl rsa -in privkey.pem -out server.key

rm privkey.pem

openssl req -x509 -in server.req -text -key server.key -out
server.crt

chmod 600 server.key
```

Server Configuration (cont.)

- **Update `pg_hba.conf`**

```
hostssl all             all             0.0.0.0/0               md5
```

- **Update `postgresql.conf`**
  - Ensure `listen_addresses` is set correctly
  - Add `ssl = on`
  - Check SSL certificate files location

```
ssl_cert_file = 'server.crt'
ssl_key_file = 'server.key'
```

- **Restart Postgres server**

Client Configuration

- **Connect using `sslmode` option with one of four values:**
  - `disable`
  - `allow`
  - `prefer`
  - `require`

PHP Connection Example

```
$link = pg_connect("host=10.4.3.2 port=5432 dbname=pgbench
user=pgbench password=pgbench sslmode=require");
```

Encrypting Data Across Network: SSH Tunnel

- No modifications to Postgres configuration
- Use of existing SSH gateway

```
ssh -f -N -L 127.0.0.1:2000:10.4.3.2:5432 user@sshgw.corp.net
```

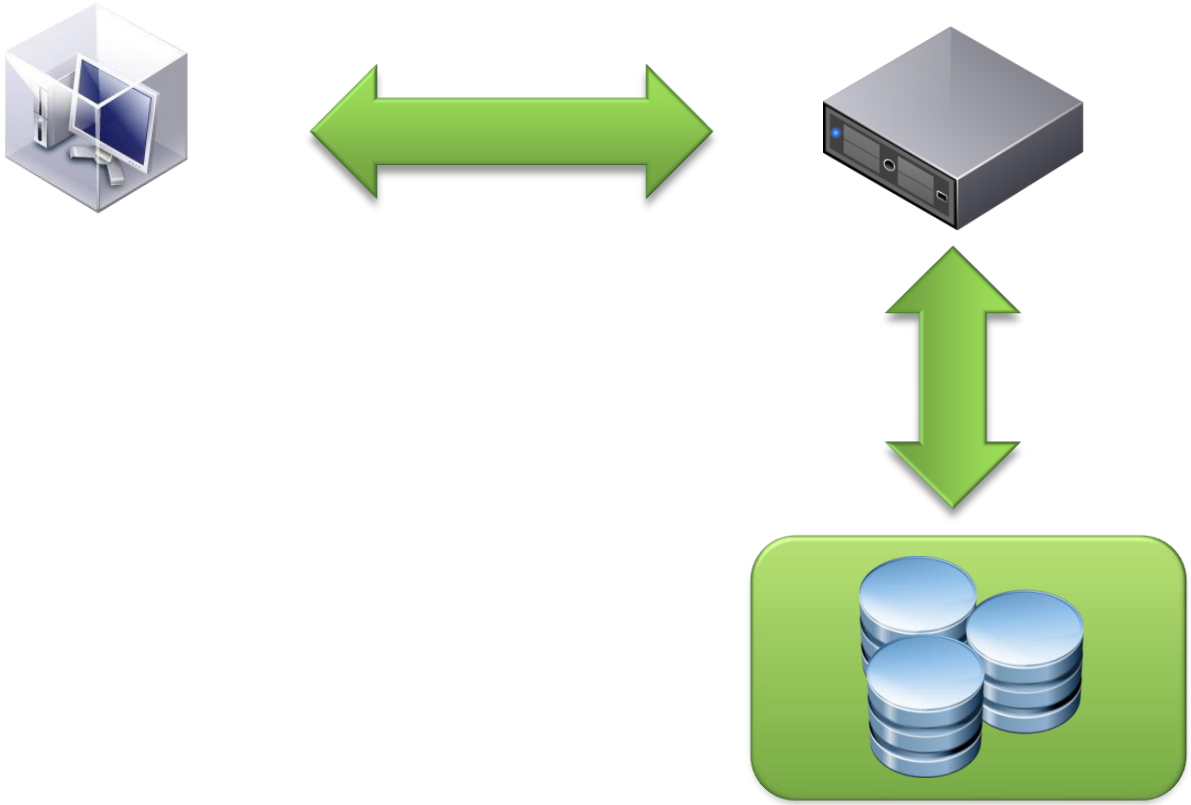
PHP Connection Example

```
$link = pg_connect("host=127.0.0.1 port=2000 dbname=pgbench
user=pgbench password=pgbench");
```

Complete Application Encryption

- **Application encrypts and writes data into database**
- **Application reads and decrypts data from database**
- **Requires no involvement of database and network**
  - Listed here for completeness
  - No tests done

Complete Client Encryption



Test Bed

- **4x Intel Xeon E5-5640 (32 cores in total), EMC VNX5500 SAN**
- **Hypervisor: VMware ESXi 5.1 Express Patch 2**
- **Virtual machine: 32 vCPUs, 12GB vRAM**
- **Guest operating system: SuSE Linux Enterprise Server 11 SP1**
- **Postgres 9.3.0:**
  - shared_buffers= 8GB, checkpoint_segments=100
  - Separate partitions for PGDATA and XLOG
- **Benchmark:**
  - pgbench -i -s 100; pgbench -c 32 -j 32 -M prepared -T 300

Encrypting Columns (pgcrypto) Tests

- **Test bed**
  - pgbench connects over LAN
  - Workload: pgbench from postgresql.git versus pgbench modified
    - pgbench modified: encrypt/decrypt `abalance` column in `pgbench_accounts`

```
UPDATE pgbench_accounts SET abalance = encrypt( decrypt(abalance) + :delta)
WHERE tid = :tid;

SELECT convert_from(decrypt(abalance, 'key'::bytea, 'aes'),
current_setting('server_encoding')) FROM pgbench_accounts WHERE aid = :aid;

UPDATE pgbench_accounts SET abalance = encrypt(0::text::bytea, 'key'::bytea,
'aes');
```

- **Results**

	Baseline	pgcrypto
pgbench tps	3483	3311

Encrypting Data Partition Tests

- **Test bed**
  - pgbench connects over Unix domain sockets
- **Results**

	Baseline	Encrypting DATA & XLOG
pgbench tps	13814	5414

Encrypting Data over Network Tests

- **Test bed**
  - pgbench connects over LAN and WAN (coast-to-coast)
- **Results**

pgbench tps	Baseline	SSL	SSH tunnel
LAN	3250	3132	1510
WAN	42.11	42.01	34.68

Real-World Use Cases

- E-commerce website
- Patient information application

E-Commerce Website

- **Case**
  - Web server is hosted on public cloud
  - Database server is hosted internally

- **Options to encrypt data on the wire**
  - SSL
  - pgcrypto for specific columns (e.g., credit card)

Patient Information Application

- **Case**
  - Internal application
  - Information remains in-house (within clinic or hospital)

- **Options to encrypt data on disk**
  - Data partition
  - Specific columns

Conclusions

- **Why Encrypt Data?**
- **Encryption Options**
  - pg_crypto and Column based Encryption
  - SSL/SSH Tunnel
  - Filesystem Encryption
- **Performance results**
- **Real-world Examples**

Questions?

References

- http://www.postgresql.org/docs/current/static/encryption-options.html
- http://www.postgresql.org/docs/current/static/pgbench.html
- http://www.postgresql.org/docs/current/static/ssl-tcp.html
- http://www.postgresql.org/docs/current/static/ssh-tunnels.html
- http://www.postgresql.org/docs/current/static/libpq-connect.html
- http://www.postgresql.org/docs/current/static/pgcrypto.html
- http://www.postgresql.org/docs/current/static/libpq-ssl.html
- http://www.revsys.com/writings/quicktips/ssh-tunnel.html
- http://cubist.cs.washington.edu/doc/ExamplePHPwPostgreSQL.shtml
- http://php.net/manual/en/ref.pgsql.php
- http://www.php.net/manual/en/function.pg-connect.php
- http://wiki.centos.org/HowTos/EncryptedFilesystem
- http://www.faqs.org/docs/Linux-HOWTO/Loopback-Encrypted-Filesystem-HOWTO.html