مستندات فنی SecuriChat

مرور کلی

SecuriChat یک برنامه پیام‌رسانی امن است که با استفاده از پایتون، Tkinter برای رابط کاربری گرافیکی، برنامه‌نویسی سوکت برای ارتباط بلادرنگ و MySQL برای ذخیره‌سازی دائمی ساخته شده است. این برنامه از ارسال پیام‌های متنی، انتقال فایل، ردیابی وضعیت کاربران و بازیابی تاریخچه گپ پشتیبانی می‌کند. همچنین شامل رمزنگاری AES برای ارتباطات امن، checksumهای CRC۳۲ برای تشخیص خطا و سیستم ثبت وقایع (Logging) جامع برای نظارت بر تمام عملیات‌هاست.

این مستند مکانیزم‌های پیاده‌سازی، از جمله معماری سیستم، پروتکل‌های ارتباطی، ویژگی‌های امنیتی، ساختار پایگاه داده، سیستم ثبت وقایع و مدیریت خطاها را شرح می‌دهد.

معماری سیستم

SecuriChat از معماری کلاینت-سرور پیروی می‌کند که شامل اجزای زیر است:

1. **کلاینت (SecuriChatGUI.py)**:
   - رابط کاربری گرافیکی مبتنی بر Tkinter که شامل صفحات ورود، انتخاب کاربر و گپ است.
   - مدیریت ارتباط سوکت با سرور برای ارسال/دریافت پیام‌ها و فایل‌ها.
   - پیاده‌سازی رمزنگاری، تأیید checksum و ثبت وقایع.
2. **سرور (server.py)**:
   - مدیریت اتصال کلاینت‌ها با استفاده از سوکت‌های TCP.
   - هدایت پیام‌ها و فایل‌ها بین کلاینت‌ها.
   - نگهداری لیست کاربران متصل و وضعیت آن‌ها.
   - ذخیره داده‌ها در پایگاه داده MySQL و ثبت وقایع.
3. **پایگاه داده (db.py)**:
   - ارتباط با پایگاه داده MySQL برای ذخیره پیام‌ها، متادیتای فایل‌ها و وضعیت کاربران.
   - پشتیبانی از بازیابی تاریخچه گپ.
4. **سیستم ثبت وقایع (logger.py)**:
   - ثبت تمام رویدادهای مهم (اتصالات، پیام‌ها، انتقال فایل‌ها، رمزنگاری، خطاها) در فایل‌های ثبت وقایع با مهر زمانی.
5. **رمزنگاری (crypto_utils.py)**:
   - ارائه رمزنگاری AES (از طریق Fernet) و checksumهای CRC۳۲ برای انتقال امن و قابل اعتماد داده‌ها.

مکانیزم‌های پیاده‌سازی

۱. ارتباط کلاینت-سرور

- **پروتکل**: استفاده از سوکت‌های TCP برای ارتباط قابل اعتماد و اتصال‌محور.
- **فرمت پیام**: ارتباط مبتنی بر هدرهای JSON-encoded و به دنبال آن داده‌های payload (برای فایل‌ها). هدرها شامل متادیتا مانند file_size، file_name، content، target، action، type و checksum هستند.
- **جریان کار**:
  - کلاینت‌ها به سرور در آدرس ۱۲۷.۰.۰.۱:۱۲۳۴۵ متصل شده و نام کاربری خود را ارسال می‌کنند.
  - سرور یک دیکشنری (clients) برای نگاشت نام‌های کاربری به اشیاء سوکت نگهداری می‌کند.
  - پیام‌ها و فایل‌ها با هدر JSON ارسال شده و برای فایل‌ها، داده‌های رمزنگاری‌شده دنبال می‌شوند.
  - سرور پیام‌ها/فایل‌ها را به کلاینت مقصد هدایت می‌کند (در صورت آنلاین بودن) یا آن‌ها را در پایگاه داده ذخیره می‌کند.

۲. رابط کاربری (Tkinter)

- **صفحه ورود**:
  - کاربران نام کاربری خود را وارد کرده و به سرور متصل می‌شوند.
  - بررسی می‌کند که نام کاربری خالی نباشد و تلاش‌های اتصال را ثبت می‌کند.
- **صفحه انتخاب کاربر**:
  - نمایش لیست کاربران آنلاین (🟢) و آفلاین (⚪) همراه با مهر زمانی آخرین بازدید.
  - امکان تازه‌سازی لیست کاربران و شروع گپ.
- **صفحه گپ**:
  - نمایش تاریخچه گپ و پیام‌های بلادرنگ.
  - پشتیبانی از ورودی متن، پیوست فایل (از طریق دکمه 📎) و دانلود فایل.
  - استفاده از ScrolledText برای نمایش گپ و Entry برای ورودی پیام.

۳. ویژگی‌های امنیتی

- **رمزنگاری**:
  - **کتابخانه**: استفاده از Fernet از کتابخانه cryptography (رمزنگاری متقارن AES).
  - **تولید کلید**: کلید با استفاده از PBKDF۲HMAC با الگوریتم SHA۲۵۶، salt ثابت (securichat_salt) و ۱۰۰,۰۰۰ تکرار، از رمز پیش‌فرض (securichat_key) تولید می‌شود.
  - **فرآیند**: پیام‌ها و فایل‌ها قبل از ارسال در کلاینت رمزنگاری شده و در زمان دریافت رمزگشایی می‌شوند. سرور نیز فایل‌ها را برای ذخیره‌سازی و ارسال رمزنگاری/رمزگشایی می‌کند.
  - **پیاده‌سازی**: کلاس CryptoUtils در فایل crypto_utils.py عملیات رمزنگاری/رمزگشایی را مدیریت می‌کند.

- **تشخیص خطا**:
  - **Checksum**: استفاده از CRC۳۲ برای تأیید صحت داده‌ها.
  - **فرآیند**: checksum قبل از رمزنگاری محاسبه شده و در هدر پیام ارسال می‌شود. گیرنده پس از رمزگشایی checksum را تأیید می‌کند. در صورت عدم تطابق، خطا ثبت و نمایش داده می‌شود.
  - **پیاده‌سازی**: متدهای calculate_checksum و verify_checksum در کلاس CryptoUtils.

۴. مدیریت پایگاه داده

- **پایگاه داده**: MySQL (دیتابیس securichat).
- **جداول**:
  - user_status:
    - username (VARCHAR، کلید اصلی): شناسه منحصربه‌فرد کاربر.
    - last_seen (DATETIME): مهر زمانی آخرین فعالیت کاربر.
  - messages:
    - id (INT، AUTO_INCREMENT، کلید اصلی): شناسه منحصربه‌فرد پیام.
    - sender (VARCHAR): نام کاربری فرستنده.
    - receiver (VARCHAR): نام کاربری گیرنده.
    - content (TEXT): محتوای پیام (رمزنگاری‌شده برای پیام‌های متنی) یا نام فایل.
    - message_type (VARCHAR): نوع پیام (text، image، video، audio، file).
    - file_path (VARCHAR): مسیر فایل ذخیره‌شده (برای پیام‌های غیرمتنی).
    - file_size (BIGINT): اندازه فایل.
    - timestamp (DATETIME): زمان ایجاد پیام.
- **عملیات‌ها**:
  - **ذخیره پیام**: ذخیره پیام‌های متنی رمزنگاری‌شده یا متادیتای فایل‌ها (save_message در db.py).
  - **بازیابی تاریخچه**: بازیابی تاریخچه گپ بین دو کاربر، مرتب‌شده بر اساس زمان (get_chat_history در db.py).

۵. سیستم ثبت وقایع

- **پیاده‌سازی**: کلاس ChatLogger در فایل logger.py از ماژول logging پایتون استفاده می‌کند.
- **ذخیره‌سازی لاگ**:
  - لاگ‌ها در پوشه logs با نام‌های فایل مانند chat_log_YYYYMMDD_HHMMSS.log ذخیره می‌شوند.
  - هر فایل لاگ از فرمت مهر زمانی برای ردیابی آسان استفاده می‌کند.
- **انواع لاگ**:
  - **اتصال**: اتصال/قطع اتصال کاربر، شروع سرور، درخواست لیست کاربران و غیره.

•**پیام**: ارسال/دریافت پیام‌های متنی، شامل فرستنده، گیرنده، محتوا و وضعیت رمزنگاری.

•**انتقال فایل**: ارسال/دریافت فایل‌ها، شامل نام فایل، اندازه و نوع.

•**رمزنگاری**: عملیات رمزنگاری/رمزگشایی برای پیام‌ها و فایل‌ها.

•**Checksum**: محاسبات و تأیید checksumها.

•**خطا**: خطاهای اتصال، رمزگشایی، عدم تطابق checksum، و غیره.

•**فرمت**: لاگ‌ها از فرمت ٪(asctime)s - ٪(levelname)s - ٪(message)s پیروی می‌کنند.

•**خروجی**: لاگ‌ها به فایل‌ها و کنسول نوشته می‌شوند تا نظارت بلادرنگ ممکن باشد.

۶. انتقال فایل

•**فرآیند**:

•کاربران فایل‌ها (تصاویر، ویدیوها، صوت یا سایر انواع) را از طریق دیالوگ فایل انتخاب می‌کنند.

•فایل‌ها رمزنگاری شده و checksum CRC۳۲ قبل از ارسال محاسبه می‌شود.

•سرور فایل‌ها را در پوشه server_media ذخیره کرده و اعلان‌ها را به گیرنده ارسال می‌کند.

•گیرندگان می‌توانند فایل‌ها را دانلود کنند که قبل از ذخیره در پوشه client_media رمزگشایی و تأیید می‌شوند.

•**متادیتا**: در جدول messages با file_size، file_path و message_type ذخیره می‌شود.

۷. مدیریت خطا

•**خطاهای اتصال**: با استفاده از بلوک‌های try-except مدیریت شده و از طریق messageboxهای Tkinter نمایش داده شده و ثبت می‌شوند.

•**خطاهای رمزگشایی**: رمزگشایی‌های ناموفق ثبت شده و به صورت "[Decryption failed]" در گپ نمایش داده می‌شوند.

•**عدم تطابق Checksum**: عدم تطابق‌های checksum ثبت شده و به صورت "[Checksum verification failed]" نمایش داده می‌شوند.

•**خطاهای پایگاه داده**: ثبت شده و به صورت پیام‌های سیستمی در رابط گپ نمایش داده می‌شوند.

جزئیات سیستم ثبت وقایع

سیستم ثبت وقایع برای نظارت بر عملیات‌های SecuriChat حیاتی است. جنبه‌های کلیدی عبارتند از:

•**پوشه لاگ**: logs/

•**نام‌گذاری فایل**: chat_log_YYYYMMDD_HHMMSS.log (مثال: chat_log_۲۰۲۵۰۶۰۸_۲۱۴۵۲۳.log).

•**سطوح لاگ**:

•INFO: برای اتصالات، پیام‌ها، انتقال فایل‌ها، رمزنگاری و عملیات checksum.

- **ERROR**: برای خطاها (اتصال، رمزگشایی، عدم تطابق checksum، پایگاه داده و غیره).

- **نمونه لاگ‌ها**:

```
۲۰۲۵-۰۶-۰۸ ۲۱:۴۵:۲۳,۱۲۳ - INFO - Connection - User: Alice, Status: Connected
۲۰۲۵-۰۶-۰۸ ۲۱:۴۵:۲۵,۴۵۶ - INFO - Message - Sender: Alice, Receiver: Bob, Type: text, Encrypted: True, Content: سلام، باب!
۲۰۲۵-۰۶-۰۸ ۲۱:۴۵:۲۷,۷۸۹ - INFO - Encryption - Operation: encrypt_message, Details: Message length: ۱۲
۲۰۲۵-۰۶-۰۸ ۲۱:۴۵:۲۷,۷۹۰ - INFO - Checksum - Operation: calculate, Data: سلام، باب!...., Checksum: ۱۲۳۴۵۶۷۸۹۰
۲۰۲۵-۰۶-۰۸ ۲۱:۴۵:۳۰,۰۱۲ - ERROR - Checksum - Verification failed for received message
```

- **موارد استفاده**:
  - اشکال‌زدایی (مثلاً خطاهای اتصال، رمزگشایی).
  - ممیزی امنیتی (مثلاً تأیید استفاده از رمزنگاری).
  - نظارت بر فعالیت کاربران و عملکرد سیستم.

ملاحظات امنیتی

- **کلید رمزنگاری**: پیاده‌سازی فعلی از رمز ثابت (securichat_key) برای تولید کلید استفاده می‌کند. برای تولید، موارد زیر را در نظر بگیرید:
  - کلیدهای جلسه‌ای برای هر کاربر.
  - تبادل کلید امن (مثلاً Diffie-Hellman).
  - مکانیزم‌های چرخش کلید.
- **Checksum‌ها**: CRC۳۲ برای سادگی استفاده شده اما از نظر رمزنگاری امن نیست. برای امنیت بیشتر، HMAC-SHA۲۵۶ را در نظر بگیرید.
- **پایگاه داده**: اعتبارنامه‌ها به صورت سخت‌کد شده‌اند (xatovate، ۱۲۳۱۲۳). در تولید، از متغیرهای محیطی یا خزانه امن استفاده کنید.
- **شبکه**: روی ۱۲۷.۰.۰.۱ اجرا می‌شود. برای دسترسی از راه دور، SSL/TLS را برای ارتباط سوکت پیاده‌سازی کنید.

وابستگی‌ها

- **کتابخانه‌های پایتون**:
  - tkinter: چارچوب رابط کاربری گرافیکی.
  - mysql-connector-python: دسترسی به پایگاه داده MySQL.
  - cryptography: رمزنگاری AES و تولید کلید.
  - Pillow: مدیریت تصاویر (برای عناصر رابط کاربری).
- **خارجی**: سرور MySQL.

دستورالعمل‌های راه‌اندازی

1. **نصب وابستگی‌ها**:

```
pip install mysql-connector-python cryptography Pillow
```

2. **راه‌اندازی پایگاه داده**:

```
CREATE DATABASE securichat;
USE securichat;
CREATE TABLE user_status (
    username VARCHAR(۵۰) PRIMARY KEY,
    last_seen DATETIME
);
CREATE TABLE messages (
    id INT AUTO_INCREMENT PRIMARY KEY,
    sender VARCHAR(۵۰),
    receiver VARCHAR(۵۰),
    content TEXT,
    message_type VARCHAR(۲۰) DEFAULT 'text',
    file_path VARCHAR(۲۵۵),
    file_size BIGINT,
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

3. **اجرای سرور**:

```
python server.py
```

4. **اجرای کلاینت**:

```
python SecuriChatGUI.py
```

محدودیت‌ها و بهبودهای آینده

- **مقیاس‌پذیری**: سرور از دیکشنری ساده برای مدیریت کلاینت‌ها استفاده می‌کند که برای تعداد زیاد کاربران مقیاس‌پذیر نیست. راه‌حل‌های قوی‌تر (مثلاً Redis) را در نظر بگیرید.
- **امنیت**: رمزنگاری را با کلیدهای پویا و تبادل کلید امن تقویت کنید.
- **اندازه فایل**: انتقال فایل‌های بزرگ ممکن است کند باشد؛ استریم تکه‌ای یا فشرده‌سازی را پیاده‌سازی کنید.
- **رابط کاربری**: افزودن ویژگی‌هایی مانند رسید خواندن پیام، نشانگر تایپ یا گپ‌های گروهی.
- **لاگ‌ها**: افزودن چرخش لاگ برای مدیریت فضای دیسک.

نتیجه‌گیری

SecuriChat یک پلتفرم امن و کاربرپسند برای پیام‌رسانی بلادرنگ و اشتراک‌گذاری فایل ارائه می‌دهد که با رمزنگاری قوی، تشخیص خطا و ثبت وقایع جامع همراه است. طراحی مدولار آن امکان گسترش آسان را فراهم می‌کند و سیستم ثبت وقایع جامع، شفافیت و قابلیت اشکال‌زدایی را تضمین می‌کند. این مستند به عنوان راهنمایی برای درک و نگهداری برنامه عمل می‌کند.