令和5年度

修　士　論　文

ワンタイムパスワードによる相互認証付き近距離ファイル共有アプリケーション

情報科学プログラム　計算機基礎学研究室
M223576　　立川 大雅

主指導教員・主査　准教授　北須賀 輝明
副指導教員・副査　　教授　　西村 浩二
　　副査　　教授　　中西 透
副指導教員・副査　准教授　今井 勝喜（福山大学）

令和 5 年 2 月 6 日

広島大学大学院先進理工系科学研究科先進理工系科学専攻

概要

スマートフォンの普及によって，私たちはアプリやスマートフォンの機能を駆使して情報やファイルを共有する機会が日常生活で増えてきている．例えば，Apple の AirDrop や Android のニアバイシェアは素早く簡単にファイルを共有できるため，多くの人が利用している．これらのサービスでは，誰からも受信可能な設定や連絡帳に登録されている人のデータのみを受信する設定がある．一方で，LINE や Slack などのアプリケーションが連絡手段の主流となり，連絡帳に友人の電話番号やメールアドレスを登録していない人も増えてきている．そのため，AirDrop やニアバイシェアを使用する際，誰からでも受信すると設定している人が多い可能性がある．誰からも受信すると設定し，AirDrop やニアバイシェアを多人数がいる場で使用すると，第 3 者が端末名を宛先と同じにするなどして成りすまし，情報を盗まれる可能性や間違った宛先を選択して誤送信する可能性がある．しかし，送信者はファイルを送信する前に送り先を確認する手段がない．そのため，ファイルを送信した後でしか，送り先の端末が正しかったのかを判断するタイミングがなく，成りすましや誤送信を防止できるとは言い難い．

そこで，本研究では近くにいる知人とファイル共有することを想定し，送信者と受信者間で相互認証を行うことにより，成りすましと誤送信を未然に防ぐファイル共有アプリケーションを提案し開発する．近くにいる知人とは，友人やクラスメイトなどの身近な人を指す．開発したアプリでは，初めて本アプリを使用してファイル共有する場合と初めてではない場合とで操作が違うため，場合を分けて説明を行う．初めての場合は，送信者と受信者の端末で Diffie-Hellman（DH）鍵交換方式を使用して共通鍵を生成し，各端末に保存する．生成した共通鍵を用いて，Time-based One-time Password（TOTP）方式によりワンタイムパスワード（OTP）を生成することで成りすましを防ぐ．この OTP は送信者と受信者の端末画面に表示され，本アプリを使用して初めてファイル共有する際の相互認証を行う．

初めてでない場合は，送信者と受信者は保存した共通鍵を利用して現時刻を使用して生

成した OTP1 と現時刻に整数 1 を足した時刻を使用して生成した OTP2 の 2 つを生成する．送信者は，OTP1 を受信者に送信し，受信者は，OTP2 を送信者に送信する．送信者は受信した OTP2 ，受信者は受信した OTP1 が自身で生成したものと一致すれば，相互認証が成功する．その後，送信者はファイルの選択が可能となる．ファイルを選択した後，送信者は事前に受信者端末に 2 桁の乱数を送信し，同時にその乱数を送信者端末に表示する．受信者は受信した 2 桁の乱数を画面に表示させ，送信者端末，受信者端末で表示されている乱数が一致すれば，送受信端末を確認したことになるので誤送信を防ぐ効果が得られる．その後，送信者はファイルを送信する．

ファイル送信は gRPC の Client streaming RPC を利用し，ファイルを分割して受信者に送信する．gRPC はクライアントサーバ型の通信プロトコルであり，アプリ内ではクライアントとサーバのプログラムが同時に実行されている．アプリで動作するサーバのポート番号は 50052 に設定しており，クライアントはサーバのポート番号と，サーバが動作する端末の IP アドレスを指定することで通信を行い，ファイルのやり取りを行う．IP アドレスは認証成功後に Bluetooth LE によって送信者と受信者が交換するように設計する．

Android 端末を使用して開発したアプリの実用性を検証した結果，初めてファイル共有する際の，Bluetooth LE 通信接続から検証のための OTP を画面に表示させるまでに平均で約 3.93 秒かかり，ユーザが遅いと感じる可能性がある結果となった．一方，初めてではない際の，Bluetooth LE 通信接続から OTP 検証が終わるまでの平均時間は約 2.65 秒と，高速に実行できることから，ユーザから遅いと感じさせにくい結果となった．

目次

図目次

表目次

第 1 章　はじめに

1.1　研究背景

スマートフォンの普及によって，我々は日常的にアプリやスマートフォンの機能を使ってファイル共有する機会が増えてきている．それに伴い，紙媒体での情報共有ではなく，電子文書による情報共有が一般的になってきた．具体的な共有方法として，我々の研究室では Slack や Teams，Dropbox，OneDrive などを使用している．

近くにいる人とファイル共有する方法として，iOS の AirDrop や Android のニアバイシェア（Nearby Share）が普及している．これら 2 つのサービスでは，誰からも受信するモードと連絡帳に登録している人のデータのみを受信するモードがある．一方，LINE や Slack などのアプリケーションが若者の間で主流に使われている時代の背景から，連絡帳に友人の電話番号やメールアドレスを登録していない人も多い．総務省情報通信政策研究所の報告書 [1] によると，LINE は多くの世代で使用されており，とりわけ 10 代から 50 代で利用率が 90% を超え，60 代は 80% を超えていることから，電話番号やメールアドレスを交換する時代から，LINE の ID を交換する時代へと変化しているのではないかと推測される．そのため，AirDrop やニアバイシェアを使う際，設定を誰からも受信すると設定している人が多いのではないかと考えられる．誰からも受信すると設定し，AirDrop やニアバイシェアを人が大勢いるところで使うと，第三者がデバイス名を宛先と同じにするなどして成りすまして情報を盗まれる可能性や間違った宛先を選択し誤送信してしまう可能性がある．しかし，送信者は選択したデバイスまたは接続先のデバイスを確認する手段がなく，ファイルを送信する前に送信先を明示的に確認する手順がない．そのため，送信者はファイルを送信する前に，受信者を選び間違っていないことを注意深く確認せねばならない．

そこで，Bluetooth LE と TOTP を利用して成りすましや誤送信を防ぎ，gRPC で近くにいる知人とファイル共有を行うためのアプリケーションを，Flutter フレームワークにより開発する．提案アプリケーションと Slack や Teams 等のファイル共有アプリケー

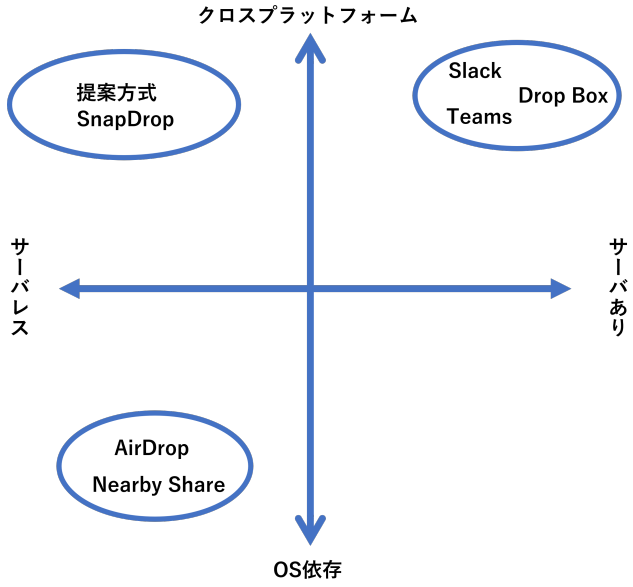


図 1.1: OS 依存性/サーバ有無マトリクス

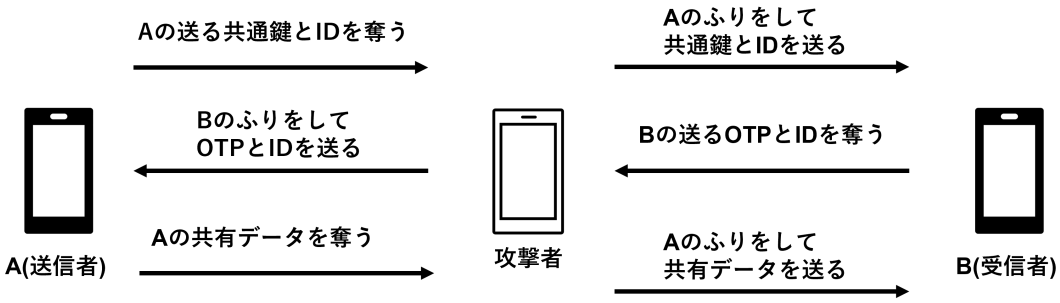


図 1.2: 成りすまし

ションの関係図を図 1.1 に示す．

成りすまし

成りすましを図 1.2 に示す．図 1.2 で攻撃者は，A に対しては B に成りすまし，B に対しては A に成りすましている．図 1.2 のように，攻撃者が A や B を装って A と B の通信へ不正に割り込む攻撃手法がある．この際，攻撃者は A が B にのみ送信する予定であった情報と，B が A にのみ送信する予定であった情報を盗み出す．

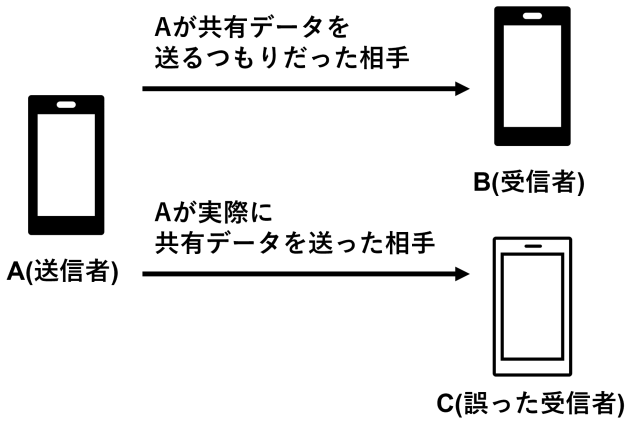


図 1.3: 誤送信

誤送信

誤送信を図 1.3 に示す．誤送信とは，本来 A は B に対してデータを送るつもりが，A 自身によるヒューマンエラーによって，受信者の選択を誤り，意図しない C に誤って送信される状況を指す．

中間者攻撃と誤送信はどちらも情報漏洩の危険性がある．

1.2　本論文の構成

本論文は全 6 章で構成される．第 2 章では，関連研究について説明する．第 3 章では，提案アプリケーションに用いる技術について説明する．第 4 章では，提案アプリケーションについて説明する．第 5 章では，開発環境や動作確認について述べる．第 6 章では，結論と今後の課題についてまとめる．

第 2 章　関連研究

2.1　AirDrop

Apple の AirDrop [2] は 近くにある iPhone や iPad, Mac 同士で画像や書類を共有できるサービスである．Bluetooth と Wi-Fi の両方を利用して共有を行う．共有データを受取る側は，AirDrop の受信設定を「すべての人」,「連絡先のみ」,「受信しない」の 3 つのうちいずれかを選び設定する．また受信者は，共有データを受信するたびに，受け入れるか拒否するかを選択できる．その際，送信者には，受信者が受け入れたか拒否したかが分かるようになっている．下記に AirDrop の操作手順を示す．

1. 送信者はアプリを開いて共有ボタンをタップ
2. AirDrop ボタンをタップ
3. 表示されている AirDrop ユーザの中から，共有相手のデバイスをタップ
4. 共有完了

AirDrop は，受信設定によるフィルタリング機能がある．また，共有される前に間違えてタップしたデバイスをもう一度タップすると送信が取り消しされる機能が備わっている．以上の機能等で送信者の誤送信を防いでいると考える．

2.2　ニアバイシェア

Android のニアバイシェア（Nearby Share）[3] は近くにある Android OS 6.0 以降のデバイス同士と写真やファイルなどを共有できる機能である．Bluetooth, Wi-Fi, 位置情報を利用して共有を行う．Wi-Fi 設定はオンにする必要があるが，Wi-Fi のない環境でも使用可能となっている．利用者は，Nearby Share の設定を「全ユーザ対象」,「連絡先」,「あなたのデバイス」の 3 つのうちから選び設定する．下記に Nearby Share の操作手順を示す．

1. 送信者はアプリを開いて共有をタップ
2. ニアバイシェア ボタンをタップ
3. 表示されている近くのスマホの中から，共有相手のデバイスをタップ
4. 共有完了

Nearby Share の特徴として，受信者が Nearby Share を利用しないと設定している場合でも，送信者が特定のデバイスを指定せず，付近のデバイスに対して共有を行うと近くにある受信者端末に「付近のデバイスが共有中です」というポップアップが表示され，ポップアップをタップすると一時的に受信者の受信設定が全ユーザ対象となり，受信することができる．

Nearby Share は Windows 用の Nearby Share [4] をインストールしている Bluetooth 対応の PC にも共有できる．そのため，Android デバイスと Windows のデスクトップおよびノートパソコンの間での写真やファイルの共有が可能となっている．

Nearby share も Airdrop と同様に，受信設定によるフィルタリング機能がある．また，送信先が誤っていた場合でも，相手が承認するまでは送信をキャンセルできる機能が備わっている．以上の機能により送信者の誤送信を防いでいると考えられる．

2.3　Musubi

Dodson ら [5] は，第三者の介入を必要とせず，暗号化したメッセージを送りあうモバイルソーシャルアプリケーションプラットフォームである「Musubi」を提案している．Musubi では通信時，認証局や他の仲介者を利用しない分散型の仕様上，ユーザの端末で暗号化する必要があるが，ユーザにとって暗号化を意識せずに使用できる仕組みとなっている．特に公開鍵の交換の動作はユーザにとって敷居が高い動作であるため，身近な動作に対応させることでユーザに思考を強いることなく自然に交換することが可能となる．具体的には NFC [6]（Near Field Communication）と呼ばれる端末同士を接触，あるいはかざすだけで通信できる技術を使用する．NFC に対応していない端末の際は，公開鍵を埋め込んだ QR コードによって，公開鍵を交換する．公開鍵の交換後にアドレス帳に登録し，取得した公開鍵の持ち主を友達と定義する．そして，アドレス帳に登録されている公開鍵の持ち主である友達とネットワークに接続することなく通信することができ，友達以外からのメッセージは即座に破棄される．

第 3 章　予備知識

3.1　Flutter

Flutter [7] は、2017 年に Google によって開発およびサポートされているオープンソースフレームワークである．Flutter はマルチプラットフォームであり，Android・iOS・Web・Windows・macOS・Linux の 6 つのプラットフォームでのアプリケーション開発をサポートしている [8][9]．そのため，モバイル以外のプラットフォームのアプリも単一のコードで開発することができる．Flutter のメリットは大きく 2 つある．

1. クロスプラットフォームに対応
   Flutter は同一のコードで iOS と Android の両方に共通した描画を表現できる機能が備わっているため，開発コストを大幅に削減できる．
2. ホットリロードに対応
   ホットリロードとは，プログラムを書き換えた際，即座に UI に反映されることである．そのため，開発中に変更点をすぐに確認でき，開発効率が高い．

3.2　Bluetooth Low Energy

Bluetooth Low Energy (LE) [10] は 2009 年 12 月 (正式リリースは 2010 年 7 月 6 日) に Bluetooth Special Interest Group (SIG) によって，Bluetooth コア仕様バージョン 4.0 に初めて搭載された機能である [11]．Bluetooth LE は，低消費電力に特化しており，従来の通信規格である Bluetooth Classic (または BR/EDR) と互換性がない．そのため，Bluetooth Classic のみに対応する機器であることを示す Bluetooth，Bluetooth LE のみに対応する機器であることを示す Bluetooth SMART，両方に対応することを示す Bluetooth SMART READY の 3 つのブランド名が存在する．

Bluetooth LE のデバイスには，セントラル（Central）とペリフェラル（Peripheral）

表 3.1: Attribute 値

値	説明
Handle	連番のインデックス. 値の範囲は 0x0001 から 0xffff まで.
Type	Service と Characteristic を示す.
Value	各属性が持つデータのこと.
Permission	読み書きの権限.

の 2 種類の役割がある．セントラルは親側の機器の名称で Bluetooth LE 通信の中央制御を担当する役割を担い，通信できる範囲に存在するペリフェラルデバイスをスキャンし，そのデバイスに接続を確立した後，データを要求・取得する．ペリフェラルは子側の機器の名称で，セントラルに対してデータを提供する役割を担う．セントラルが同時に接続できるペリフェラルの数は，仕様上は無限だがセントラルのデバイス性能によって制限がある [12].

次に Bluetooth LE の接続について説明する．図 3.4 はセントラルとペリフェラルの接続を表している．まず，ペリフェラルは近くにいるセントラルに対して接続を要求するアドバタイジングパケットを，接続が完了するまで 20 ミリ秒から 10.24 秒の間隔でブロードキャストする [13]．このアドバタイジングパケットをブロードキャストする通信方法をアドバタイジングと呼ぶ．セントラルはスキャンをすることでアドバタイジングパケットを受信し，接続したい場合は接続要求を送信する．ペリフェラルは接続要求を受信した後，アドバタイジングをやめ，接続が完了という流れになる [14]．データのやり取りを行い，セントラルが接続を切断するとペリフェラルはアドバタイジングを再開する．

3.2.1　Attribute Protocol

Attribute Protocol（ATT）はクライアント・サーバアーキテクチャを実現するためのプロトコルである．通常は接続が完了すると，セントラルはクライアントの役割，ペリフェラルはサーバの役割となり，クライアント・サーバ関係が形成される．クライアントとサーバはそれぞれ Attribute（属性）データを所持しており，Attribute をやり取りするプロトコルが Attribute Protocol である．Attribute は表 3.1 の 4 つの値で構成される．

Handle は，属性の識別子であり通信の際に属性を特定するために使用される．具体的

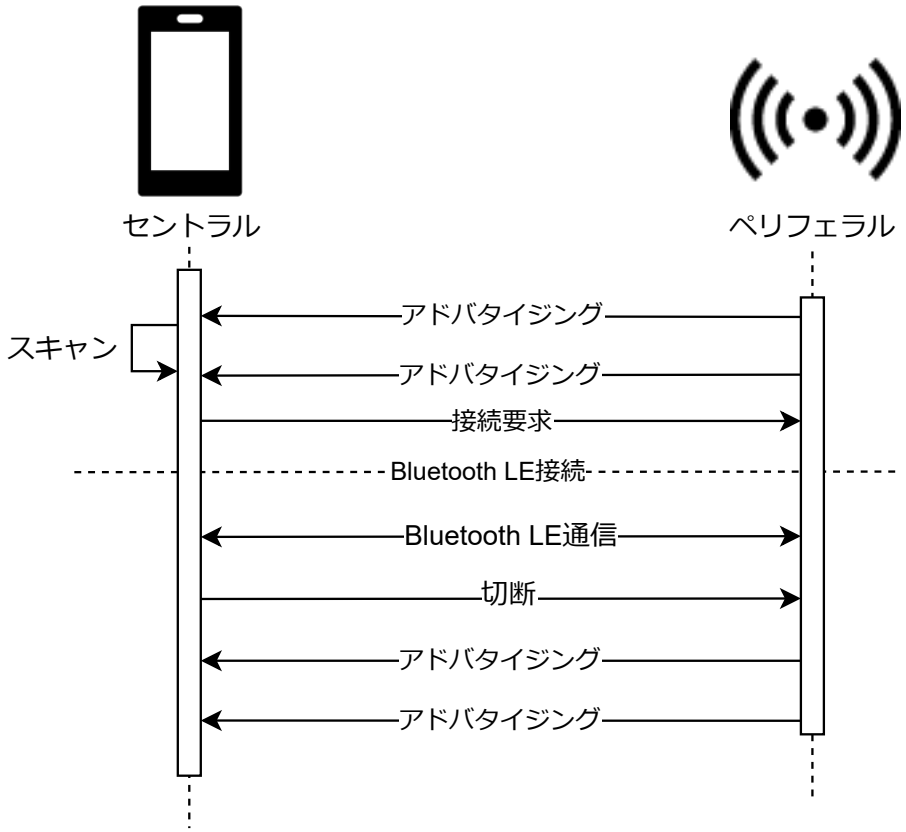


図 3.4: Bluetooth LE の接続

には，クライアントがサーバに対して Read や Write などの操作を行う際に，対象となる属性を明示的に指定するために使用される．Type は属性種類を示す．これによって，クライアントがサーバに対してどの属性にアクセスするかを特定できる．Value は，各属性が持つデータのことを指し，Service とその中に含まれる属性である Characteristic によって表現する．Permission は，読み取り，書き込みのどちらか，あるいはその両方が許可されているかを表す．

本研究では，読み取りと書き込みの両方を許可している．

3.2.2　Generic Attribute Profile

Generic Attribute Profile（GATT）は，ATT を介して Service（サービス），Characteristic（特性） のデータ型を使用する一連の手順を定義する．サーバは図 3.5 に示すように Service でグループ化された Characteristic をクライアントに公開する [15].

GATT サーバで公開される特性には表 3.2 に示すプロパティが設定され，サーバ上で

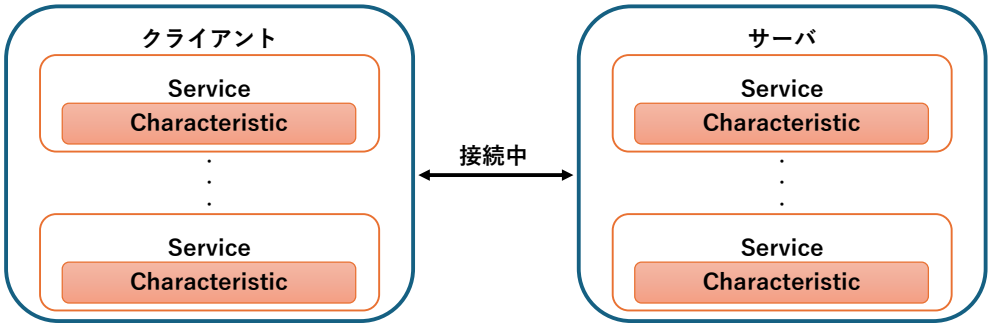


図 3.5: Generic Attribute Profile（GATT）

表 3.2: 特性プロパティ

プロパティ	説明
Broadcast	サーバがデータをブロードキャストする.
Read	クライアントからデータの読み出しが可能.
Write Without Response	サーバからのレスポンスの要求なしで, クライアントからのデータの書き込みが可能.
Write	サーバからのレスポンスの要求ありで, クライアントからデータの書き込みが可能.
Notify	クライアントからのレスポンスの要求なしで，サーバ からクライアントへ characteristic の変更を通知する.
Indicate	クライアントからのレスポンスの要求ありで，サーバ からクライアントへ characteristic の変更を通知する.
Authenticated Signed Writes	クライアントからデータの署名付き書き込みが可能.
Extended Properties	拡張プロパティ

公開されるデータのやり取りを行うために，以下に示す手続きが定義されている.

- クライアントからサーバのサービスの検索
- クライアントからサーバの特性の検索
- クライアントからサーバのデータの読み出し（Read）
- クライアントからサーバのデータの書き込み（Write）
- サーバからクライアントへデータの表示を行いクライアントからサーバへの受信の

確認を行う（Indication）

- サーバからクライアントへデータの通知（Notification）

本研究では，Read と Write プロパティを使用し，Bluetooth LE によって通信を行う．

3.3 Time-based One-time Password

Time-based One-Time Password（TOTP）[16] とは，現在の時刻と共有鍵 K を用いてワンタイムパスワード (OTP) を生成する方式のことである．TOTP のデフォルトでは 30 秒ごとに 6 桁の新しい OTP を生成する．式 (3.1) で OTP は生成される．

$$\begin{aligned} \mathrm{TOTP}(K,T) &= \mathrm{Truncate}(\text{HMAC-SHA-1}(K,T)) \\ T &= \lfloor (\text{現在の Unix 時間} - T0)/X \rfloor \end{aligned} \tag{3.1}$$

ここで，Truncate はユーザが簡単に入力できるように，指定した桁数の OTP を生成する関数である．HMAC-SHA-1 アルゴリズムは RFC 2104 [17] で定義されたもので，160 ビット (20 バイト) を生成する．T は TOTP を生成するためのタイムステップを示し，$T0$ はタイムステップのカウントを開始する Unix 時間を示す．$T0$ はデフォルトでは 0 になっている．K は共通鍵を示す．最後に X は TOTP を再生成する秒数で 30 秒がデフォルトになっている．

本研究では，送信者と受信者の相互認証を行う際に，Time-based One-Time Password を使用する．

3.4 Diffie-Hellman 鍵交換方式

Diffie と Hellman [18][19][20] は，初めて暗号で離散対数問題を用いた Diffie-Hellman (DH) 鍵交換方式と呼ばれている方式を発見する．離散対数問題とは，ある群 $\mathbb{G}$ の上で定義される．群 $\mathbb{G}$ の離散対数問題とは，g を位数が素数 p であるような $\mathbb{G}$ の要素，$y = g^x \bmod p$ ($x \in \{0,1,2,...,p-1\}$，以降 $\{0,1,2,...,p-1\}$ を $\mathbb{Z}_p$ とする) としたとき，(g,y,p) が与えられて，x を求める問題のことである．しかし，群が大きい場合，x を求めることは非常に困難であるとされている．

DH 鍵交換方式での鍵交換を図 3.6 に示す．送信者 A と受信者 B を用いて説明する．まず，A はランダムに $x \in \mathbb{Z}_p$ を選び，$A = g^x$ を計算し，B に送る．B も同様にランダムに $y \in \mathbb{Z}_p$ を選び，$A = g^y$ を計算し，A に送る．A は B との共通鍵 K とし，$K_A = (g^y)^x$ を計算し，B は $K_B = (g^x)^y$ を計算する．この時，$K_A = (g^y)^x = g^{xy} = (g^x)^y = K_B$ と

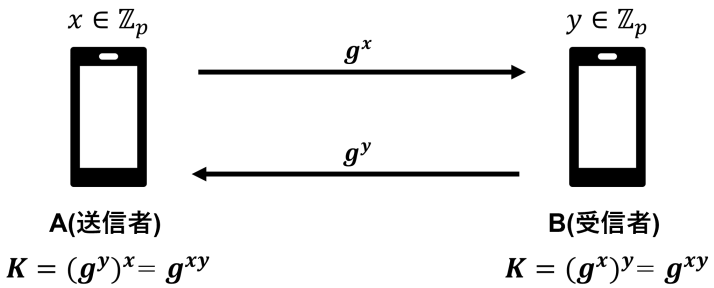


図 3.6: DH 鍵交換方式

なるので，A と B 間で同じ鍵を共有できる．この際，A と B の通信を第三者 M が盗聴していたとしても，M は g^x と g^y を入手できるが，この 2 つから共通鍵 g^{xy} を計算することは，困難である．理由として，g^{xy} を計算するは，g^x と g^y のどちらか一方から x や y を求めなければならない，つまり，離散対数を解く必要がある．しかし，離散対数は解くことが非常に困難であると予想されている．この予想は「Diffie-Hellman（DH）仮定」と呼ばれている．この DH 仮定から盗聴するだけの攻撃には安全と言える．

本研究では，Time-based One-Time Password を生成する際に使用する共通鍵 K を DH 鍵交換方式を用いて共有した．

3.5　gRPC

gRPC [21][22] は，Google が開発したオープンソースの RPC （Remote Procedure Call）フレームワークのことである．RPC [23] とはクライアントサーバ型の通信プロトコルであり，サーバ上で実装されている関数をクライアントからの呼び出しに応じて実行する技術を指す．RPC では，クライアントがサーバに対して，実行する処理を指定するパラメータや引数として与えるデータを送信し，サーバはパラメータに応じた処理を実行してその結果をクライアントに返す．

gRPC では RPC と同様にクライアントサーバ型の通信プロトコルを採用しており，特定の言語やプラットフォームに依存しないようマルチプラットフォームに設計されている．そのため，様々な言語向けにクライアントとサーバが異なる言語で実装されている場合でも，通信が可能となっている．通信には HTTP/2 が，データのシリアライズには Protocol Buffers という技術がデフォルトで使用されるようになっている．Protocol Buffers は Google が開発したデータフォーマットで，バイナリデータを含むデータでも効率的に扱える．

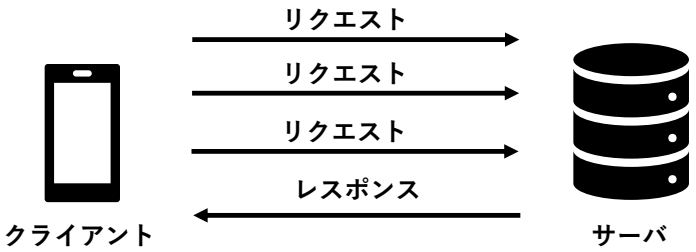


図 3.7: Client streaming RPC

本研究では，gRPC の Client streaming RPC を用いてファイル共有を行う．gRPC では，4 つの通信方法があり，その中の一つである Client streaming RPC について説明する．Client streaming RPC は図 3.7 のように，クライアントがサーバに複数回のリクエストを送信し，サーバはクライアントのリクエストに対して 1 度だけレスポンスを返す通信方法である．使用用途として，大きなサイズのファイルや多くのファイルをサーバに送信したい際に，クライアントはデータを複数回に分けてサーバに送信し，サーバは全て受信した段階で受信したことを返すといったこと例が挙げられる．

Client streaming RPC の他に，Unary RPC，Server streaming RPC，Bidirectional streaming RPC といった通信方法がある．

第 4 章　提案アプリケーション

4.1　提案アプリケーションの概要

本研究では，近くにいる知人と Android 端末間でファイル共有を行う際，送信者（セントラル）と受信者（ペリフェラル）の成りすましや誤送信を防止するために，送信者と受信者を TOTP によって認証する機能が付いたアプリケーションの開発を行う．

提案アプリと従来のファイル共有アプリの異なる点を説明する．従来アプリでは，送信者は受信者を認証せずにファイル共有を行えたため，オートマトンで表すと図 4.8 のように条件分岐のないオートマトンになる．しかし，提案アプリは，図 4.9 のように従来のアプリに 4 状態を加えたオートマトンに表すことができる．加えた 4 状態を以下に示す．

1. 端末の登録チェック
   アプリ内に受信者の端末情報が登録してあるかチェックを行う状態．登録済みの場合は以前ファイル共有したことのある端末で，未登録の端末は初めて本アプリでファイル共有を行う端末.
2. 共通鍵の生成と OTP の表示
   受信者の端末情報が未登録の場合に，互いに共通鍵と共通鍵を用いてワンタイムパスワード（OTP）を生成し，OTP を送信者と受信者の端末上に表示させて，両者の認証を行う状態.
3. 共通鍵を登録
   認証が成功した場合に，これからファイル共有を行うと判断し，アプリ内に受信者の情報と共通鍵をペアで登録する状態.
4. 2 桁の乱数
   受信者の端末情報が登録済みの場合に，登録してある情報を用いて認証を行った後，ファイルを送信する前に送り先を間違えていないかの最終確認として送信者と受信者の端末上に 2 桁の乱数を表示させる状態.

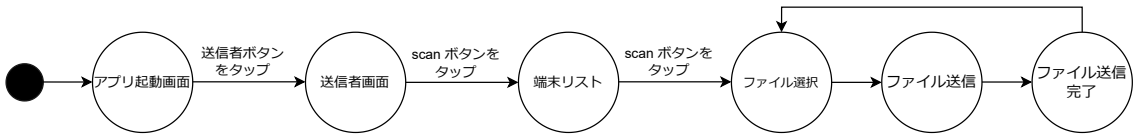


図 4.8: 従来のファイル共有手順

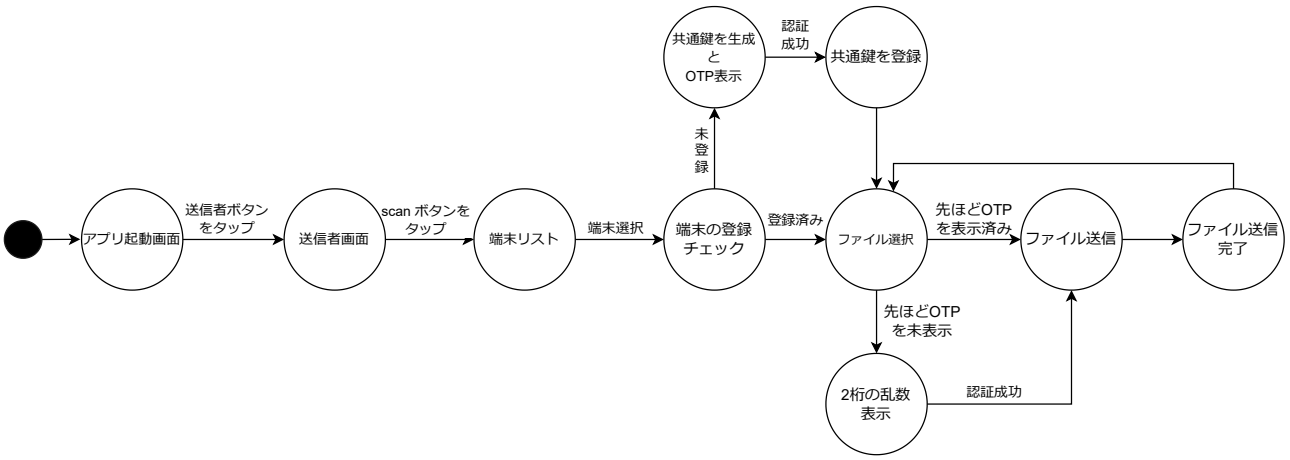


図 4.9: 提案アプリのファイル共有手順

送信者は OTP と 2 桁のによって認証を行うことで成りすましや誤送信を防ぐことができる．また，認証に必要な情報は，Bluetooth LE 通信によって送信を行い，認証成功後のファイル共有は，gRPC を使ってファイル共有を行う．以上が本アプリの概要となる．

4.2　成りすまし・誤送信の防止認証

成りすましの防止認証と誤送信の防止認証について説明する．成りすましと誤送信の認証は，初めてファイル共有する際と 2 回目以降にファイル共有する際とで手順が変わってくるため，場合を分けて説明を行う．

初めてファイル共有する場合

初めてファイル共有する場合は，DH 鍵交換方式 [20] を用いて，認証で使用する共通鍵を交換する．しかし，DH 鍵交換は成りすまし攻撃に対しては，安全ではない．図 4.10 に例を示す．A と B の通信に攻撃者 M が割って入り，A と M の間では，共通鍵 K_1 を共有し，B と M の間では，K_2 を共有する．この時，A は B と，B は A と通信している

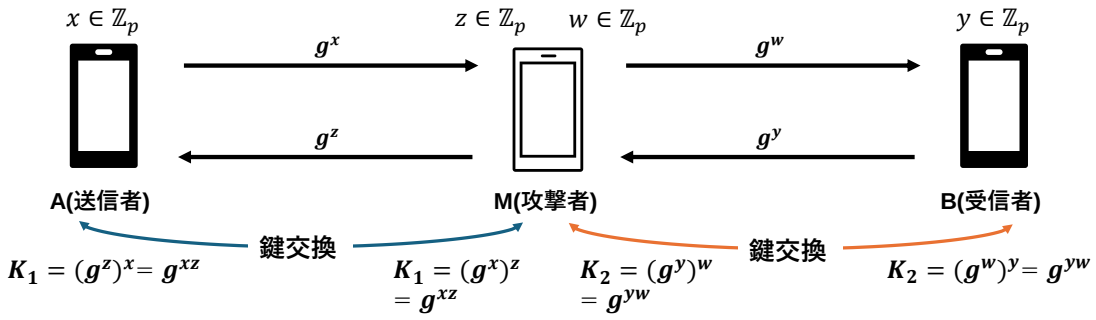


図 4.10: 成りすましされている時の DH 鍵交換

と信じているため，A は K_1 を用いて B に対して暗号通信を行い，B は K_2 を用いて A に対して暗号通信を行う．しかし，M は K_1 と K_2 を所持しているため，通信内容はすべて M に盗聴されていることになる．そこで，交換した共通鍵を使って 6 桁の OTP を生成し A と B の端末画面に表示させ，お互いに数値が同じかを確認する．6 桁の OTP は式 (3.1) を使用して生成する．この手順により，先ほど A の通信相手が B であることと，B の通信相手が A であることが確認できるため，成りすましと誤送信が防げる．

M からの成りすましを防止できる理由として，図 4.10 のような M が成りすましている状況を仮定した際，A は式 (3.1) の K の部分に K_1 を代入した TOTP(K_1, T) で，B は K の部分に K_2 を代入した TOTP(K_2, T) で OTP を生成することになり，A と B の OTP が一致しない．このことから，A と B は共通鍵が同じでないことが分かる．

一方で，OTP が一致してる際は，同じ共通鍵 $K = g^{xy}$ で生成していると判断できるため，図 3.6 のように DH 鍵交換が A と B の間で行われ，通信・鍵交換相手が A は B，B は A であると分かる仕組みになっている．認証が終わると，A は B の固有の ID と共通鍵 K を，B は A の ID と共通鍵 K をペアでアプリ内に保管しファイル共有が行える．これらのことから，OTP を生成し見せあうことで成りすましと誤送信が防げる．

2 回目以降ファイル共有する場合

2 回目以降ファイル共有する場合は，1 回目のファイル共有時に，交換した共通鍵を使用して 6 桁と 2 桁の乱数を生成し認証を行う．

まず 6 桁の OTP をほぼ同時に共有し，相手を確認する．6 桁の OTP を共有する際，A と B はほぼ同じ時間に OTP を生成するため，式 (3.1) をこのまま使用すると，同じ OTP である TOTP(K, T) を生成し共有することになる．図 4.11 のように，同じ値の

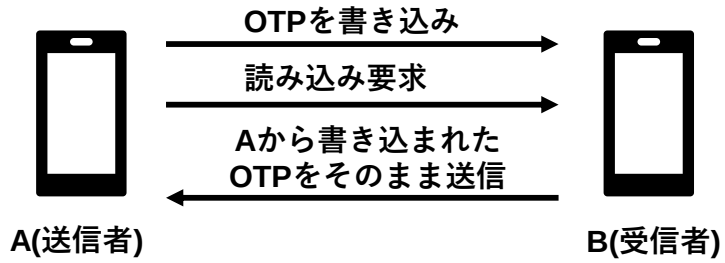


図 4.11: 2 回目以降ファイル共有する際の認証の問題

OTP を共有し合った場合に認証を成功としてしまうと，A が B に送った後に，B は A から送られてきた OTP を送り返すだけで認証が成功してしまい安全とは言えない．そこで，B は式 (4.2) を使用して OTP 生成し A に共有することとした．式 (4.2) は式 (3.1) の T に整数 1 を足した T' を使用して OTP を生成する式となる．A はあらかじめ式 (4.2) を使用して OTP を生成しておき, B から送られた OTP を認証する．一方, A は式 (3.1) を使用し OTP を生成して B に送る．B もあらかじめ式 (3.1) を使用して OTP を生成しておき，A から送られた OTP を認証する．以上をまとめたものを表 4.3 に示す．

6 桁の OTP 認証が成功すると，B の端末に式 (3.1) で生成した 2 桁の OTP を表示させ，A は 2 桁の数値を自身の端末に入力し，A の端末があらかじめ生成していた数値と一致した場合，認証成功となる．成功すると，gRPC によるファイル共有ができる．

DH 鍵共有を利用して鍵交換された共通鍵と式 (3.1)，式 (4.2) を併用することで，成りすましと誤送信が防止できる．

$$\begin{aligned} \mathrm{TOTP}(K, T') &= \mathrm{Truncate}(\text{HMAC-SHA-1}(K, T')) \\ T' &= \lfloor (\text{現在の Unix 時間} - T0)/X \rfloor + 1 \end{aligned} \tag{4.2}$$

表 4.3: まとめ

	送信時	受信した OTP を検証時
送信者 A	$\mathrm{TOTP}(K, T)$	$\mathrm{TOTP}(K, T')$
受信者 B	$\mathrm{TOTP}(K, T')$	$\mathrm{TOTP}(K, T)$

4.3　ファイル共有

ファイル共有は，gRPC によって実装を行う．アプリ内ではクライアントの役割のプログラムとサーバの役割のプログラムが同時に実行されている．アプリで動作するサーバのポート番号を 50052 に指定しており，クライアントはサーバのポート番号と，サーバが動作する端末の IP アドレスを指定することで通信を行い，サーバにファイルの送信を行う．

ファイル送信は gRPC の Client streaming RPC により，ファイルを分割して B に送信される．最初にファイル名を送信し，その以降に送信されるデータが実際のファイルデータになる．ファイルは 2 MB ごとに分割され，それぞれのチャンクが個別に送信される．また，40 秒以内に全てのチャンクが送信されない場合はタイムアウトとなる．

Client streaming RPC によって分割され送信されるデータは，最初に送られてきたファイル名と，それ以降に送られてくるファイルデータであるチャンクを逐次的に結合していき，全てのチャンクを結合し終えるとファイルとしてダウンロードフォルダに保存する．

4.4　提案アプリケーションの詳細

送信者とファイル共有相手である受信者に分けて提案アプリケーションの詳細について説明する．

4.4.1　送信者の詳細

初めての場合

アプリ起動後，送信者 A は「送信者」ボタンを選択し送信者画面に遷移する．送信者画面には，「scan」ボタンがありタップすると，他端末からブロードキャストされているアドバタイジングをスキャンする．スキャンは 15 秒で自動的に停止する．スキャンが完了するとアドバタイジングしている端末名のリストと端末名の横に 「CONNECT」ボタンが表示される．「CONNECT」ボタンをタップすると，Bluetooth LE 接続が行われる．A は接続したい受信者 B のデバイス名を探して，「CONNECT」ボタンをタップし，Bluetooth LE 接続を試みる．Bluetooth LE 接続が完了すると，A 端末は B の端末に ID を送信し，アプリ内に ID と共通鍵が保存されていないことを確認する．確認後，Write Without Response プロパティによって，GATT サーバ（B 端末）に共通鍵の一

部である g^x を書き込み，Read プロパティによって，B 端末に読み取りを要求し共通鍵の一部である g^y を送信してもらう．送信してもらった g^y から共通鍵 g^{xy} を生成し，その共通鍵を使って OTP を生成する．生成された OTP は画面に表示されるので，その OTP と B の端末に表示されている OTP が一致していれば，OTP と同時に表示される「はい」ボタンをタップし，一致していなければ「いいえ」ボタンをタップする．「はい」ボタンをタップすると，B 端末の ID と共通鍵をペアでアプリ内に保存し，画面が遷移する．遷移した画面に「ファイル選択」ボタンと「送信」ボタンがあるため，「ファイル選択」ボタンをタップして送信したいファイルを選択し，「送信」ボタンをタップすると B にファイルが送信され完了となる．

2 回目以降の場合

2 回目以降も初めての時と「CONNECT」ボタンをタップして，Bluetooth LE 接続するところまでは一緒である．接続が完了すると，A 端末は接続中の B 端末に ID を送信し，ID からアプリ内に共通鍵が保存されていることを確認する．確認後，Write Without Response プロパティによって，式 (3.1) で生成した OTP を書き込み，Read プロパティによって，B から式 (4.2) で生成された OTP を読み込む．読み込んだ OTP と事前に式 (4.2) で生成していた OTP が一致していれば，「ファイル選択」ボタンと「送信」ボタンがある画面に遷移する．送りたいファイルを選択して，「送信」ボタンをタップすると，2 桁の乱数を B 端末に書き込み，A，B の画面に表示される．A，B の画面に表示されている乱数が一致していれば，正しい宛先だと確認できるため，乱数と同時に表示される「はい」ボタンをタップし，一致していなければ「いいえ」ボタンをタップする．「はい」ボタンをタップすると，ファイルが送信され完了となる．

4.4.2　受信者の詳細

初めての場合

アプリ起動後，受信者 B は「受信者」ボタンを選択して受信者画面に遷移し，B はアドバタイジングを開始する．遷移後，A が「CONNECT」をタップして Bluetooth LE 接続が完了すると，6 桁の OTP が画面表示されるので，表示された OTP を A に知らせる．また，A の画面に表示された OTP を聞き，一致していれば，OTP と同時に表示される「はい」ボタンをタップし，一致していなければ「いいえ」ボタンをタップする．その後，A からファイルを受信すると，受信したこと A に通知して終了となる．

表 4.4: 送信者 A のアプリケーション動作

	初めてファイル共有をする際	2 回目以降にファイル共有をする際
OTP	式 (3.1) で生成した OTP を画面表示	式 (3.1) で生成した OTP を受信者 B に 書き込む
2 桁の乱数	表示しない	ファイル送信時に表示する
OTP の検証	Bluetooth LE 接続後に画面表示される OTP（式 (3.1) で生成）と B の端末に表示 される OTP が一致するか送信者 A が検証	Bluetooth LE 接続後に式 (4.2) で 生成した OTP と B から送信された OTP が一致するか A の端末が検証

表 4.5: 受信者 B のアプリケーション動作

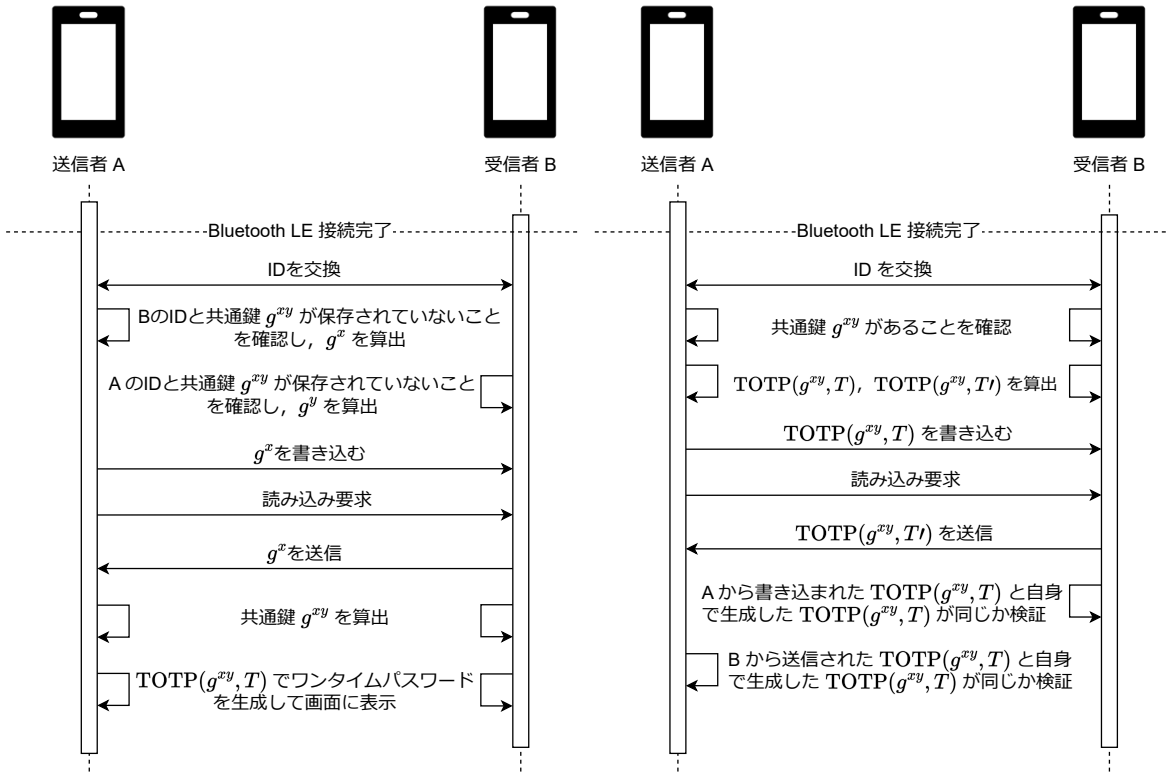
	初めてファイル共有をする際	2 回目以降にファイル共有をする際
OTP	式 (3.1) で生成した OTP を画面表示	式 (4.2) で生成した OTP を送信者 A に送信する
2 桁の乱数	表示しない	ファイル送信時に表示する
OTP の検証	Bluetooth LE 接続後に画面表示される OTP（式 (3.1) で生成）と A の端末に表示 される OTP が一致するか B が検証	Bluetooth LE 接続後に式 (3.1) で生成 した OTP と A から書き込まれた OTP が一致するか B の端末が検証

2 回目以降の場合

2 回目以降はも初めての時と同じく,「受信者」ボタンを選択して受信者画面に遷移する．遷移後に，A が「CONNECT」をタップして Bluetooth LE 接続が完了すると，OTP は表示されずにファイル共有の画面に遷移する．遷移後，A がファイルを送信しようとすると，2 桁の乱数が書いてあるポップアップが A と B の画面に表示されので，その乱数を A に伝える．A は自身の画面に表示されている乱数と B の画面に表示されている乱数が一致していれば，B にファイルが送信され完了となる．

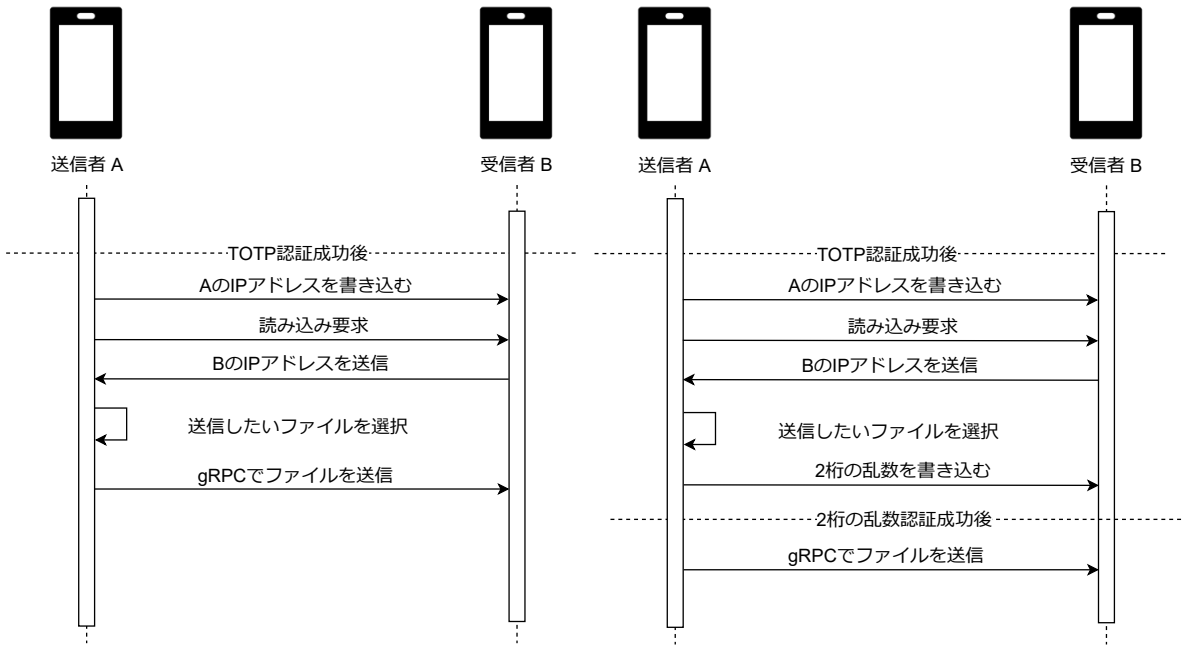
4.4.3　まとめ

表 4.4 と表 4.5 に提案アプリケーションの動作についてまとめる．また，本アプリの認証フローを図 4.12a と図 4.12b に，送信フローを図 4.13a と図 4.13b に示す．



(a) A と B が初めてファイル共有する時　　(b) 初めてではない時

図 4.12: 認証フロー



(a) A と B が初めてファイル共有する時　　(b) 初めてではない時

図 4.13: 送信フロー

第 5 章　実装

5.1　開発環境

表 5.6 に示す開発環境で提案アプリケーションを実装した．また，Android 端末である Pixel 7（Android バージョン 14，API レベル 34）にアプリをインストールし，動作を確認した．API レベルとは，Android バージョンを一意に識別する値で，Android システムとの通信に使用されている．

アプリケーションの Bluetooth LE のセントラル部分を flutter_blue_plus 1.20.8 [24]，ペリフェラル部分を ble_peripheral 0.0.1 [25]，DH 鍵交換方式を cryptography 2.7.0 [26]，ID と共通鍵をペアで保存する部分を flutter_secure_storage 9.0.0 [27] ライブラリを使用した．また，ファイル転送部分は Flutter_FileTransfer [28] OSS を使用した．

表 5.6: 開発環境

OS	windows 11 home
CPU	Intel(R) Core(TM) i5-10400 CPU @ 2.90GHz
IDE	Android Studio Giraffe — 2022.3.1 Patch 1
フレームワーク	Flutter 3.13.0
開発言語	Dart
メモリ	15.7 GB
ライブラリ	flutter_blue_plus 1.20.8 ble_peripheral 0.0.1 cryptography 2.7.0 flutter_secure_storage 9.0.0
ファイル送信に使用した OSS	Flutter_FileTransfer

5.2　アプリ設計と操作手順

図 5.14 と図 5.15 が送信者のアプリ画面，図 5.16 と図 5.17 が受信者のアプリ画面である．

アプリの操作手順

送信者端末の操作手順を **Algorithm 1**，受信者端末の操作手順を **Algorithm 2** に示す．

送信者の手順

送信者はまず，図 5.14a の「送信者」ボタンを選択すると，scan と書かれているボタンのある画面 (図 5.14b) に遷移する．「scan」ボタンをタップすと，近くにいる受信者画面を開いた端末名リストが表示される (図 5.14c)．そのリストから自分が通信したい相手を選び，「CONNECT」ボタンをタップする．「CONNECT」ボタンをタップすると，接続相手 (受信者) と初めてファイル共有する際は OTP と「はい」ボタン，「いいえ」ボタンがあるポップアップが表示され (図 5.14d)，はいをタップするとファイル共有画面 (図 5.15a) に遷移し，「いいえ」をタップするとポップアップが消えファイル共有画面には遷移しない．以前接続相手とファイル共有をしたことがある場合は，「CONNECT」ボタンをタップするとすぐファイル共有画面に遷移する．ファイル共有画面では，「ファイル選択」ボタンと「送信」ボタンがある．「ファイル選択」ボタンをタップすると端末内のファイルを選ぶことができ，送信したいファイルを選択できる (図 5.15b)．ファイルを選んだ後に，「送信」ボタンをタップすると相手にファイルが送信される．しかし，以前接続相手とファイル共有をしたことがある場合は，「送信」ボタンをタップすると，2 桁の乱数と「送信」ボタン，「いいえ」ボタンが表示される (図 5.15c)．送信ボタンを選択するとファイルが送信され，いいえボタンを選択すると送信されない．送信が成功すると，自身の画面に「転送が完了しました」と表示され (図 5.15d)，失敗すると「転送に失敗しました」と表示される．

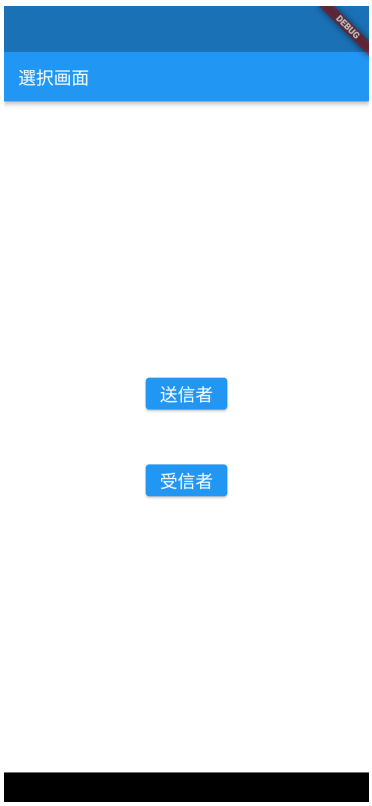
上記で述べた，送信者のアプリ画面の遷移関係を図 5.18 に示す．

受信者の手順

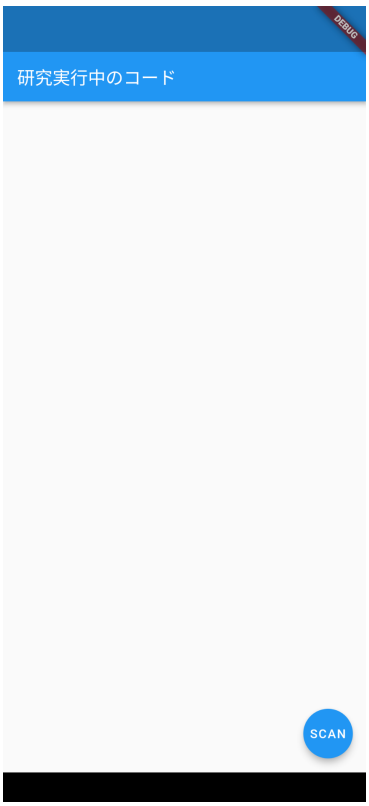
受信者はまず，図 5.16a の「受信者」ボタンを選択すると，受信者画面と書かれている画面 (図 5.16b) に遷移する．その後，送信者が CONNECT ボタンをタップすると，送

信者と初めてファイル共有する際は，OTP と「はい」ボタン，「いいえ」ボタンが表示される (図 5.16c)．はいボタンをタップすると，ファイル共有画面 (図 5.16d) に遷移して受信者もファイル共有を行うことができる．一方で，いいえボタンを押すと受信者画面にとどまり，ファイル共有ができない．送信者と以前ファイル共有していた際は，送信者が CONNECT ボタンをタップした数秒後にファイル共有画面へと遷移する．その際，送信者からファイルが送られようとすると，2 桁の乱数が表示されたポップアップ (図 5.17a) がでるため，送信者へと提示する．2 桁の乱数認証が成功し，送信者から送られたファイルの受信に成功すると，ファイルを受信しましたと表示される (図 5.17b)．

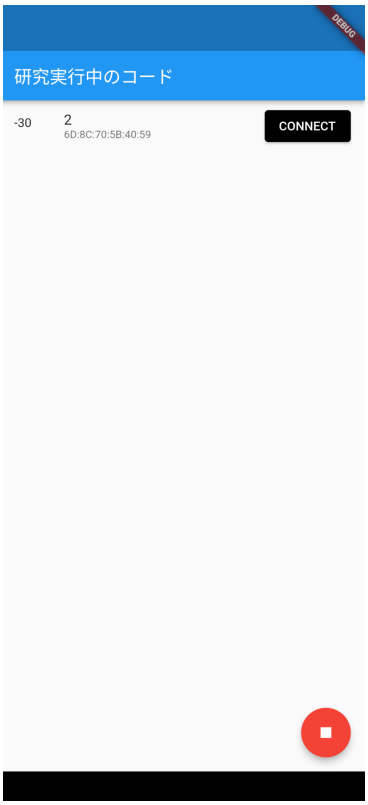
上記で述べた，受信者のアプリ画面の遷移関係を図 5.19 に示す．



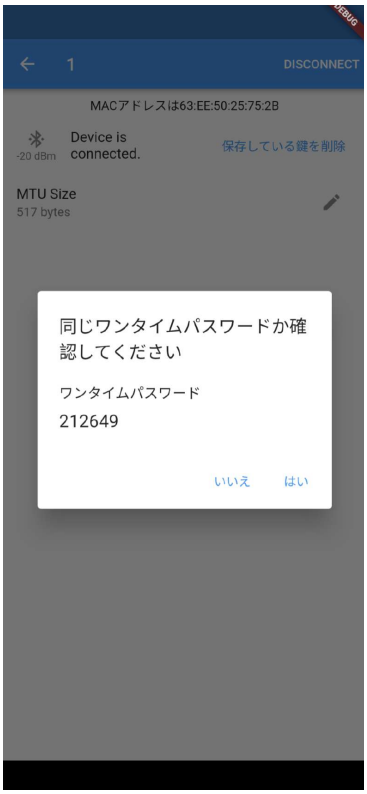
(a) 画面選択



(b) 送信者画面



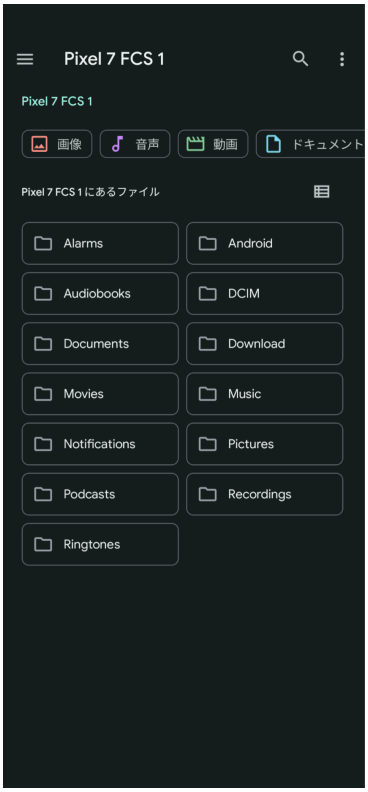
(c) アドバタイジングスキャン



(d) 初めてファイル共有する際の OTP

図 5.14: 送信者の画面 1

(a) ファイル共有画面



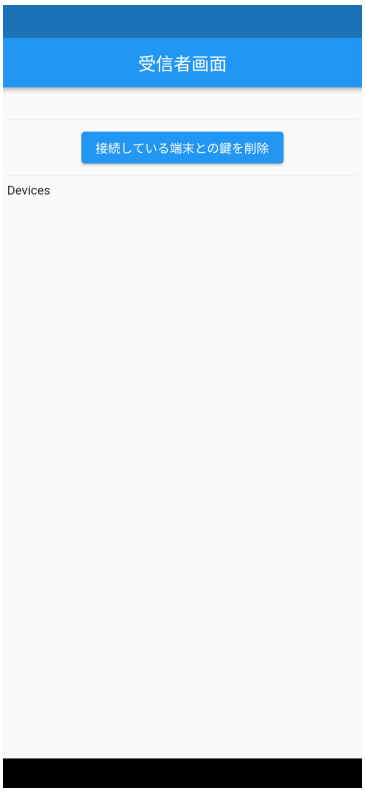
(b) ファイルを選択

(c) 2 桁の乱数

(d) 送信成功

図 5.15: 送信者の画面 2

(a) 画面選択

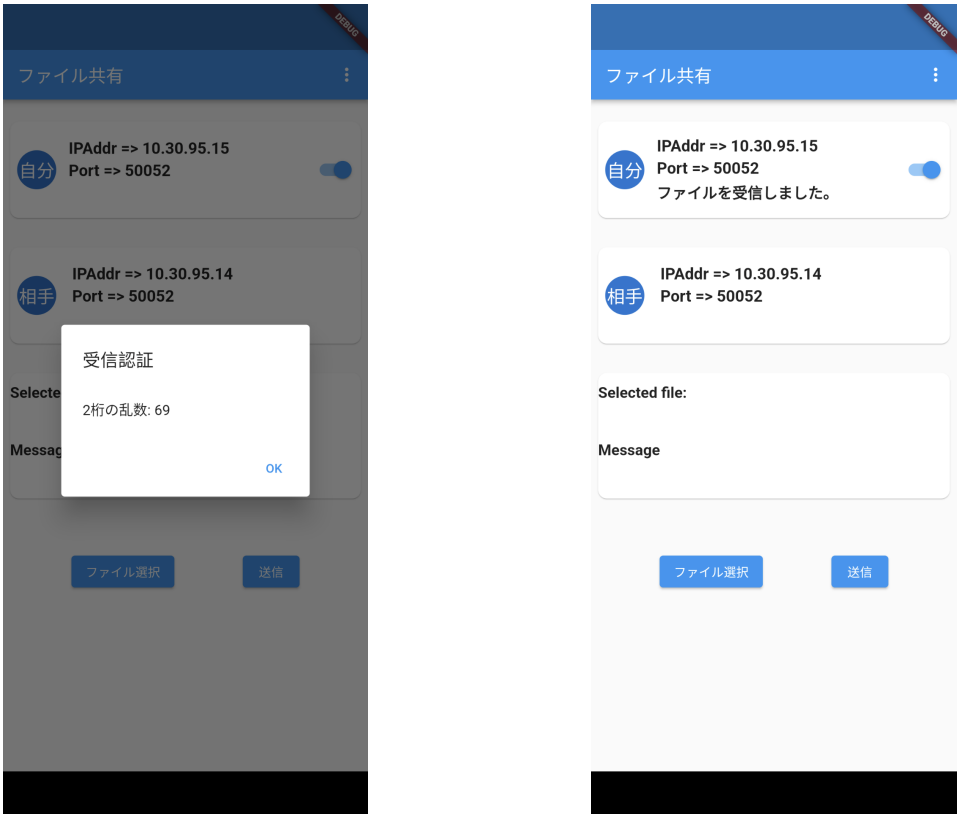


(b) 受信者画面

(c) 初めてファイル共有する際の OTP

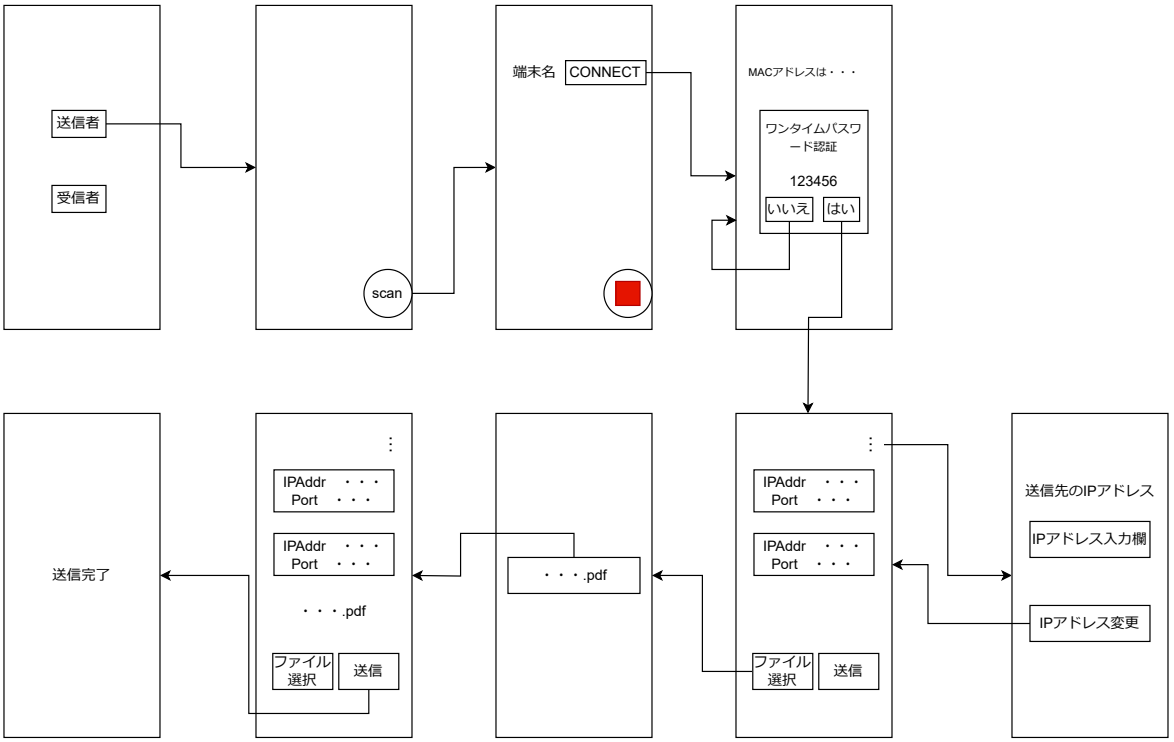
(d) ファイル共有画面

図 5.16: 受信者の画面 1

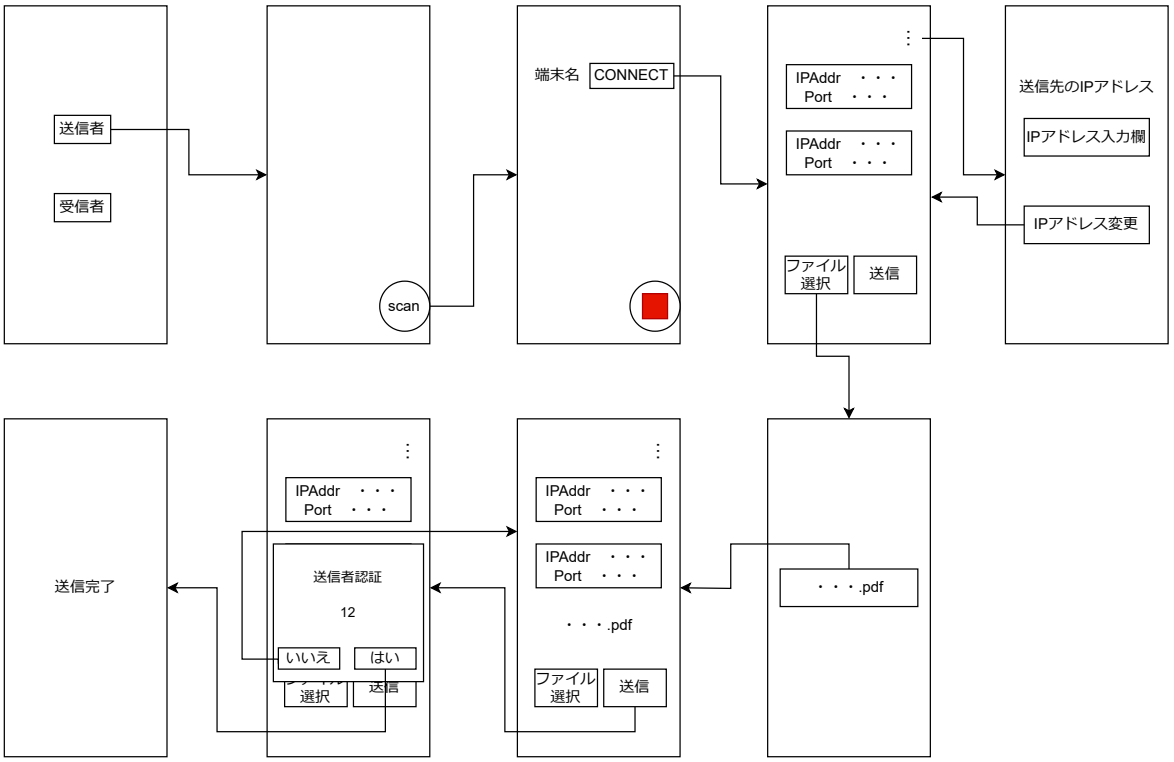


(a) 2 桁の乱数　　(b) 受信成功

図 5.17: 受信者の画面 2

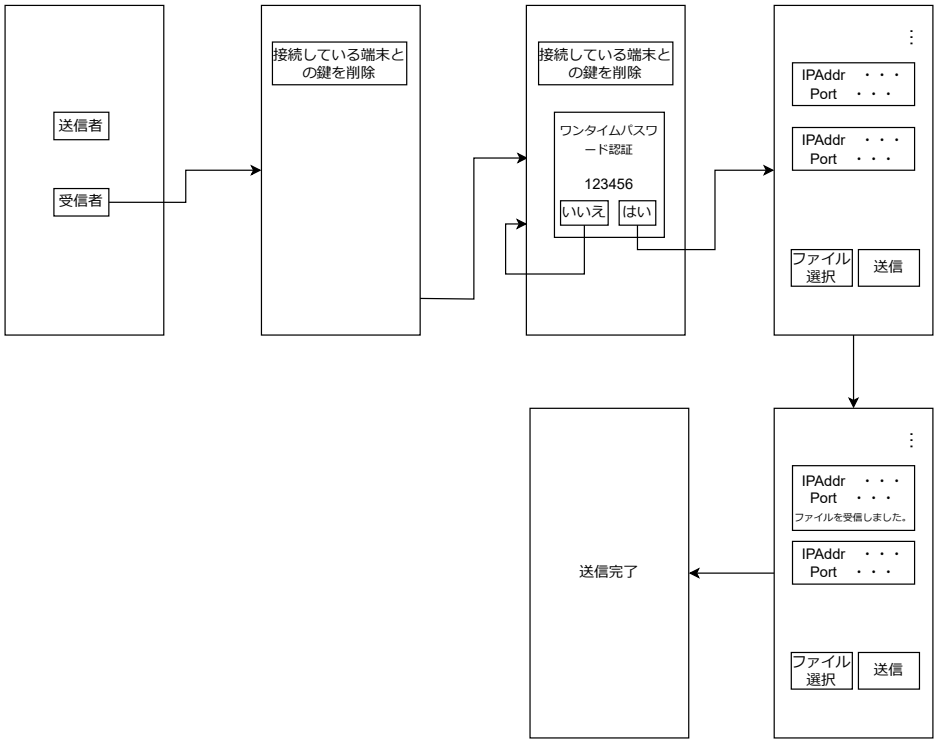


(a) A と B が初めてファイル共有する時の A の画面遷移図

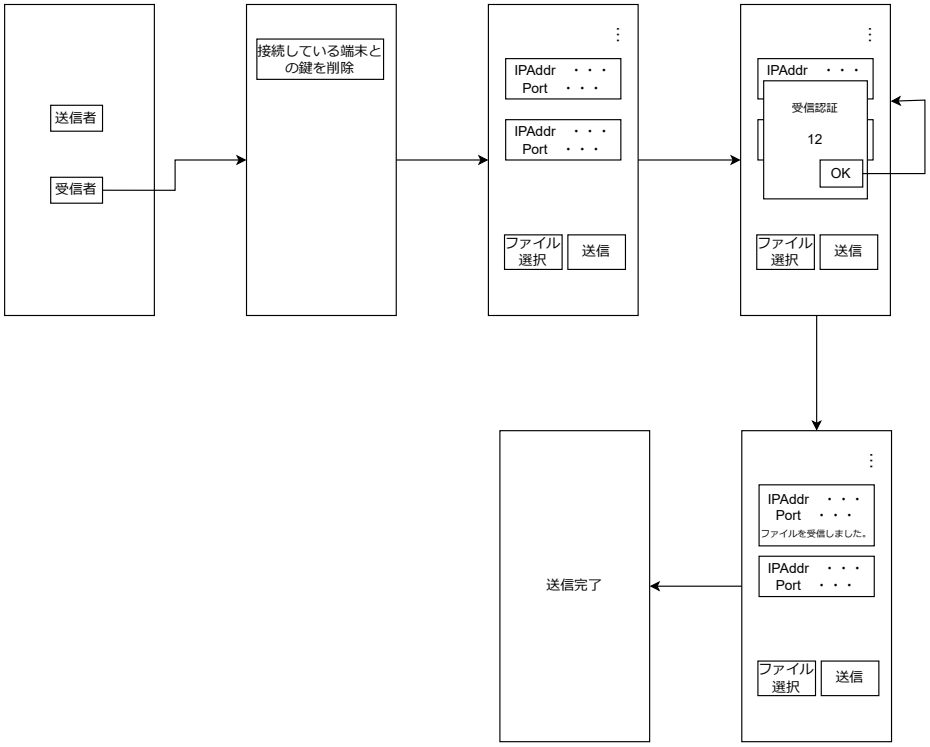


(b) A と B が 2 回目以降ファイル共有する時の A の画面遷移図

図 5.18: 送信者の画面遷移



(a) 送信者 A と受信者 B が初めてファイル共有する時の B の画面遷移図



(b) A と B が 2 回目以降ファイル共有する時の B の画面遷移図

図 5.19: 受信者の画面遷移

Algorithm 1 送信者の手順

1: アプリを起動し「送信者」ボタンを選択
2: 「scan」ボタンをタップ
3: 表示された受信者の端末名横の「CONNECT」ボタンをタップ
4: **if** 受信者と初めて本アプリでファイル共有を行う **then**
5:　画面表示された 6 桁の OTP を受信者に提示
6:　**if** 受信者と OTP が一致 (相互認証成功) **then**
7:　　「はい」ボタンを選択
8:　　受信者の IP アドレスを入力
9:　　「ファイル選択」ボタンをタップして送りたいファイルを選択
10:　　「送信」ボタンをタップ
11:　　ファイル共有開始
12:　　受信者が受取れたかを受信
13:　**end if**
14: **else**
15:　受信者の IP アドレスを入力
16:　「ファイル選択」ボタンをタップして送りたいファイルを選択
17:　「送信」ボタンをタップ
18:　受信者端末に表示された 2 桁の乱数を確認
19:　**if** 受信者と OTP が一致 (相互認証成功) **then**
20:　　「はい」ボタンを選択
21:　　ファイル共有開始
22:　　受信者が受取れたかを受信
23:　**end if**
24: **end if**
25: **if** 相互認証失敗 **then**
26:　「いいえ」ボタンを選択
27:　ファイル共有失敗
28: **end if**

Algorithm 2 受信者 B の手順

1: アプリを起動し「受信者」ボタンを選択
2: アドバタイジングを開始
3: **if** 送信者 A と初めて本アプリでファイル共有を行う **then**
4:　画面表示された 6 桁の OTP を受信者に提示
5:　**if** A と OTP が一致 (相互認証成功) **then**
6:　　「はい」ボタンを選択
7:　**end if**
8: **else**
9:　自身の端末に表示されている 2 桁の OTP を A に提示
10: **end if**
11: **if** 相互認証成功 **then**
12:　ファイルを受取る
13: **else**
14:　「いいえ」ボタンを選択
15:　ファイル共有失敗
16: **end if**

第 6 章　認証時間と評価

本章では，Pixel 7（Android バージョン 14，API レベル 34）に提案アプリケーションをインストールし，その実用性を Google が公表した Web サイトの表示時間と直帰率の関係についての調査結果を基に実装時間の評価を記述する．

6.1　Google の調査

Google の 2017 年から 2018 年の調査 [29] によると，Web ページのリンクをタップしてから表示されるまでの時間と表示されるまでの待ち時間によるユーザの直帰率[*1]の関係を次のように述べている．

> ウェブページの表示に 3 秒以上かかると，53 % のユーザはページの閲覧を止める．また，ページの表示速度が 1 秒から 3 秒になると，1 秒の時と比べ，直帰率が 32 % 増加し， 5 秒まで遅くなると 90 % 増加，6 秒だと 106 % 増加，10 秒になると 123 % 増加する．

この調査から，ユーザーは画面に何らかの変化が起こると期待してタップした際，3 秒以上時間がかかるとその待ち時間を不快に感じる可能性があると推測される．従って，本研究では，認証の目標処理時間を 3 秒以内に設定する．

6.2　評価項目

本研究では，初めて提案アプリでファイルを共有する際と，初めてではない際とで認証方法が変わるため，以下の 4 つの場合の時間を計測し，ユーザの待ち時間を調べた．

1. 送信者が本アプリで初めてファイル共有する際の「CONNECT」ボタンをタップ

[*1] 直帰率：他のページには移動せず，単一ページのみでサイトを離れる割合

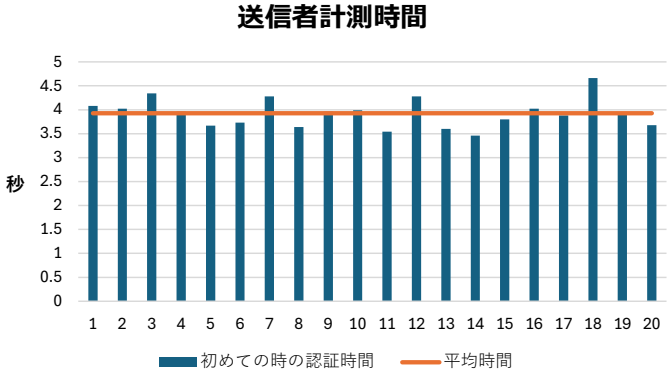


図 6.20: 送信者と受信者が初めてファイル共有する時の送信者の認証時間

して，OTP のポップアップが表示されるまでの時間[*2]

2. 送信者が本アプリで以前ファイル共有したことある際の「CONNECT」ボタンをタップして，ファイル共有画面に遷移するまでの時間[*2]
3. 受信者が本アプリで初めてファイル共有する際の送信者と Bluetooth LE 接続してから，OTP のポップアップが表示されるまでの時間[*2]
4. 受信者が本アプリで以前ファイル共有したことある際の送信者と Bluetooth LE 接続してから，ファイル共有画面に遷移するまでの時間[*2]

1．送信者が本アプリで初めてファイル共有する際の「CONNECT」ボタンをタップして，OTP のポップアップが表示されるまでの時間

表 6.7a と図 6.20 に計測時間を示す．20 回指定の動作を繰り返して計測した結果，平均時間 3.93 秒となり，目標の 3 秒を約 1 秒ほど超えてしまう結果となった．一方で，不偏分散が約 0.092 と小さいことからアプリは安定した動作を繰り返していることが分かる．そのため，ユーザに一定の期間使用してもらうことで，ユーザはポップアップが表示される時間の間隔を覚えることができ，この時間帯に表示されることに慣れてくる可能性がある．しかし，ユーザーがまだ慣れていない段階では，「タップしたのに動作しない」といった平均 3.93 秒の待ち時間により，不快感を抱く可能性が大いにあるため，認証時間の高速化が必要である．

[*2] Bluetooth LE 接続が失敗した時を除く．
今回，20 回の接続試行のうち，5 回失敗した．

表 6.7: 送信者の認証時間

(a) 初めての際

	計測時間 (秒)
1	4.08
2	4.02
3	4.34
4	3.96
5	3.67
6	3.73
7	4.28
8	3.64
9	3.96
10	3.99
11	3.54
12	4.28
13	3.60
14	3.46
15	3.80
16	4.02
17	3.88
18	4.66
19	3.95
20	3.68
平均	3.92
不偏分散	0.09

(b) 2 回目以降

	計測時間 (秒)
1	2.80
2	2.78
3	2.73
4	2.88
5	2.85
6	2.74
7	2.89
8	2.82
9	3.30
10	2.99
11	3.13
12	2.62
13	3.45
14	2.85
15	2.77
16	2.84
17	2.79
18	3.40
19	2.66
20	3.22
平均	2.92
不偏分散	0.059

2．送信者が本アプリで以前ファイル共有したことある際の「CONNECT」ボタンをタップして，ファイル共有画面に遷移するまでの時間

表 6.7b と図 6.21 に計測時間を示す．20 回指定の動作を繰り返して計測した結果，平均約時間 2.93 秒となり，目標の 3 秒を下回ることができた．また，不偏分散が 約 0.059

表 6.8: 受信者の認証時間

(a) 初めての際

	計測時間 (秒)
1	2.87
2	2.81
3	2.81
4	2.66
5	2.60
6	2.78
7	2.86
8	2.89
9	2.91
10	2.64
11	2.78
12	2.74
13	2.56
14	2.67
15	2.72
16	2.67
17	2.80
18	2.60
19	2.58
20	2.87
平均	2.74
不偏分散	0.013

(b) 2 回目以降

	計測時間 (秒)
1	2.37
2	2.39
3	2.19
4	2.34
5	2.18
6	2.25
7	2.36
8	2.11
9	2.44
10	1.92
11	2.16
12	2.13
13	2.34
14	2.26
15	2.20
16	2.07
17	2.48
18	2.31
19	2.41
20	2.21
平均	2.25
不偏分散	0.0199

と小さいことから，アプリは安定した時間で動作を繰り返していることが分かる．3 秒を超えた回数も 4 回だけだったことから，ユーザに問題なくアプリを使用してもらえると考える．

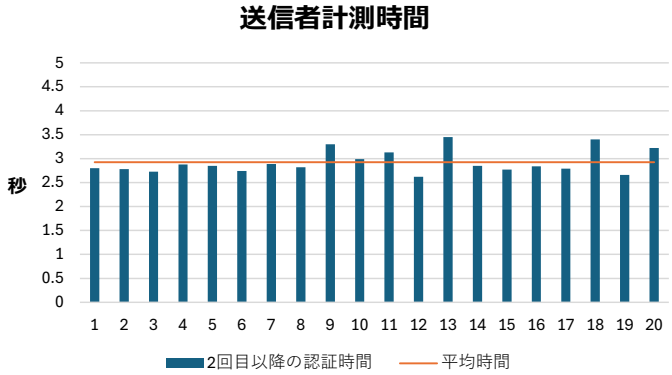


図 6.21: 送信者と受信者が 2 回目以降ファイル共有する時の送信者の認証時間

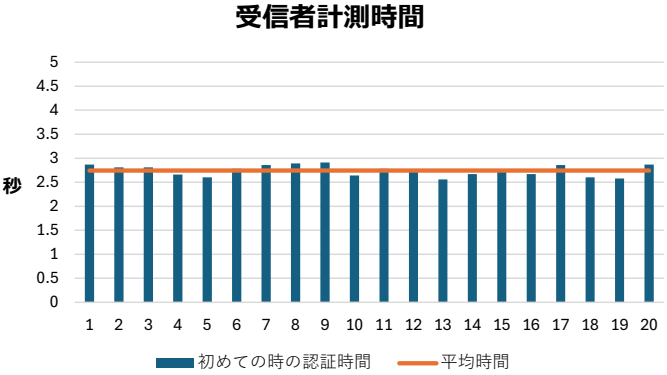


図 6.22: 送信者と受信者が初めてファイル共有する時の受信者の認証時間

3．受信者が本アプリで初めてファイル共有する際の送信者と Bluetooth LE 接続してから，OTP のポップアップが表示されるまでの時間

表 6.8a と図 6.22 に計測時間を示す．20 回指定の動作を計測した結果，平均時間 2.74 と目標の 3 秒を下回ることができた．また，不偏分散も 0.013 と小さいことから一定の時間で動作ができていることが確認できる．よって，受信者は待ち時間にそれほどまで不快感を感じないと推測する．

4．受信者が本アプリで以前ファイル共有したことある際の送信者と Bluetooth LE 接続してから，ファイル共有画面に遷移するまでの時間

表 6.8b と図 6.23 に計測時間を示す．20 回指定の動作を計測した結果，平均時間 2.26 と目標の 3 秒を下回ることができた．また，不偏分散も約 0.012 と小さいことから一定の時間で動作ができていることが確認できる．よって，実用性はあると推測する．

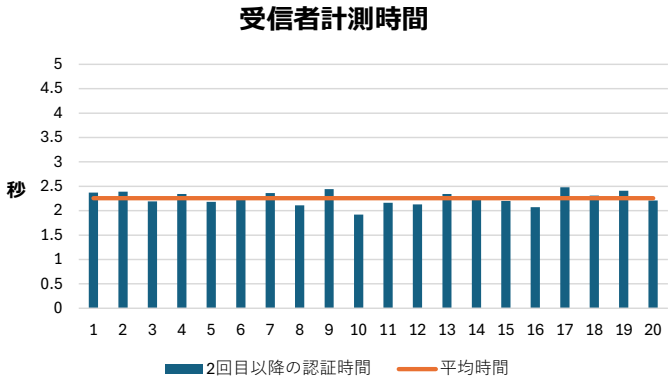


図 6.23: 送信者と受信者が 2 回目以降ファイル共有する時の受信者の認証時間

6.3　認証時間の評価

計測の結果から，2 回目以降，本アプリを使用してファイル共有する際の認証は速いものの，初めての際はユーザの待ち時間が遅いことが分かった．このことから，本アプリは使い始めの際に待ち時間の印象を与えてしまい，継続してアプリを使用してもらえない可能性がある．よって，初めてファイル共有する際に使用する認証をより高速にできる仕組みを考える必要がある．

第 7 章　おわりに

本研究では，送信者と受信者が TOTP 方式から生成されるワンタイムパスワードを用いて，相互に認証が可能となるファイル共有アプリケーションを提案し開発を行った．既存のファイル共有サービスでは，送信者が受信者を認証する方法がなかったため，成りすましや誤送信の被害に遭う恐れがあった．そこで，DH 鍵交換方式を用いて安全に送信者と受信者間で共通鍵を生成した後，共通鍵を用いてワンタイムパスワードを生成し，送信者と受信者端末に表示させることで，両者が認証できるファイル共有アプリケーションを開発した．また，生成した共通鍵を保存しておくことで，2 回目以降の認証を簡単にできる仕組みを考案した．

今後の課題としては，以下の 5 点が挙げられる．

1. 保存選択機能の追加
   開発したアプリでは，受信者が受信したファイルを強制的に保存させられているため，受信者の意思で保存するか選択できるようにする．
2. 認証時間の高速化
   ユーザにアプリの動作が遅いと感じさせないよう，認証時間をより高速化できる仕組みを考える．
3. 共通鍵の無効化機能
   古くなった共通鍵を自動で失効させる機能を追加する
4. ファイル共有する際の認証の選択
   状況に応じて，セキュアにファイルを共有する必要がある場合とそうでない場合があるため，それぞれのケースに対応できるようにアプリを改善する．
5. チャット機能の拡張
   ファイルだけでなく，チャット機能を搭載し，リアルタイムにメッセージを共有できるように拡張する．

これらが今後の課題である.

謝辞

本研究は，広島大学先進理工系科学研究科・計算機基礎学研究室で行ったものです．本研究を進めるにあたって，未熟な私に丁寧かつ温かいご指導を賜りました北須賀輝明准教授に御礼申し上げます．

また，ゼミを通じて様々な知識をご教示してくださいました中西透教授，今井勝喜准教授，加えて，本研究について多くの助言をいただきました西村浩二教授，並びに計算機基礎学研究室の大学院生及び学部生の皆様に御礼申し上げます．

参考文献

[1] 総務省情報通信政策研究所，“令和４年度情報通信メディアの利用時間と情報行動に関する調査報告書，” 2023 年 6 月.

[2] Apple Support，“iPhone や iPad で AirDrop を使う方法，” https://support.apple.com/ja-jp/HT204144，(2023/10/21 参照).

[3] Android Magazine，“近くのスマホとデータを簡単共有。ニアバイシェアの設定方法と、使えないときの対処法，” https://www.android.com/intl/ja_jp/articles/208/，(2023/10/21 参照).

[4] Android，“PC とのワイヤレス共有を簡単に実現，” https://www.android.com/intl/ja_jp/better-together/nearby-share-app/，(2023/10/21 参照)

[5] Ben Dodson, Ian Vo, T. J. Purtell, Aemon Cannon and Monica S. Lam, “Musubi: Disintermediated Interactive Social Feeds for Mobile Devices，” WWW 2012：Proceedings of the 21st international conference on World Wide Web，pp. 211–220，Apr. 2012.

[6] Hussein Ahmad Al-Ofeishat，Mohammad A.A.Al Rababah，“Near Field Communication（NFC），” IJCSNS International Journal of Computer Science and Network Security, VOL.12 No.2，Feb. 2012.

[7] Flutter，https://flutter.dev/，(2023/09/04 参照).

[8] 掛内一章，“動かして学ぶ！ Flutter 開発入門，” 翔泳社，2023 年 5 月，[560 ページ].

[9] 澤良弘，上村隆弘，村岡直人，多田幸一，“現場で使える　Flutter 開発入門，” マイナビ出版，2021 年 8 月，[384 ページ].

[10] Bluetooth Special Interest Group (SIG), https://www.bluetooth.com/learn-about-bluetooth/tech-overview/

[11] Bluetooth Special Interest Group (SIG), “Bluetooth® Low Energy Primer Document Version: 1.1.0，” Jan. 2023.

[12] 東芝情報システム株式会社, “Bluetooth とは何か？, ”https://www.tjsys.co.jp/focuson/clme-bluetooth/, (2023/09/04 参照)

[13] ルネサス エレクトロニクス株式会社, “IoT を実現する Bluetooth® Low Energy 技術とは何か, ” 2018 年 11 月, https://www.renesas.com/jp/ja/document/whp/bluetooth-low-energy-technology-brings-iot-life.

[14] 丸文株式会社, “はじめての BLE, ” https://www.marubun.co.jp/technicalsquare/9091/, (2023/09/04 参照).

[15] ルネサス エレクトロニクス株式会社, “Bluetooth® Low Energy プロトコルスタック, ” 2022 年 1 月, https://www.renesas.com/jp/ja/document/mat/bluetooth-low-energy-protocol-stack-users-manual-rev122.

[16] D.M’ Raihi, S.Machani, M.Pei, J.Rydell, “TOTP: Time-Based One-Time Password Algorithm, ” Internet Engineering Task Force, RFC: 6238, May 2011.

[17] H. Krawczyk, M. Bellare, R. Canetti, “HMAC: Keyed-Hashing for Message Authentication, ” Internet Engineering Task Force, RFC: 2104, Feb. 1997.

[18] W. Diffie, M. Hellman, “New directions in cryptography, ” IEEE TRANSACTIONS ON INFORMATION THEORY, VOL.IT-22 NO.6, Nov.1976.

[19] 岡本龍明, “鍵交換：現代暗号の誕生とその発展, ” 電子情報通信学会 基礎・境界ソサイエティ Fundamentals Review , 1 巻 4 号, 2008 年 4 月.

[20] T. Elgamal, “A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, ” IEEE Transactions on Information Theory 31.4, pp.469-472, Jul. 1985.

[21] gRPC Authors, “gRPC - A high performance, open source universal RPC framework, ” https://grpc.io/, (2024/01/14 参照).

[22] SAKURA internet, “サービス間通信のための新技術「gRPC」入門, ” https://knowledge.sakura.ad.jp/24059/#HTTPRPCgRPC, (2024/01/14 参照).

[23] A. Adamson, N. Williams, “Remote Procedure Call (RPC) Security Version 3, ” Internet Engineering Task Force, RFC: 7861, Nov.2016.

[24] “flutter_blue_plus, ”https://github.com/boskokg/flutter_blue_plus, (2024/01/14 参照).

[25] “ble_peripheral, ”https://github.com/rohitsangwan01/ble_peripheral, (2024/01/14 参照).

[26] “cryptography, ” https://github.com/dint-dev/cryptography, (2024/01/14 参照).

[27] “flutter_secure_storage, ”https://github.com/mogol/flutter_secure_storage/, (2024/01/14 参照).

[28] Keisuke Hongyo, “Flutter_FileTransfer, ” https://github.com/Keisuke-Hongyo/Flutter_FileTransfer, (2024/01/14 参照).

[29] Daniel An, “Find Out How You Stack Up to New Industry Benchmarks for Mobile Page Speed, ” Think with Google, Feb. 2017, https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/mobile-page-speed-new-industry-benchmarks/ (2024/01/14 参照).

外部発表一覧

1. Taiga Tatsukawa, Teruaki Kitasuka, Katsunobu Imai, “Method for Selecting Relay Nodes in Delay Tolerant Network Routing Using Periodicity, ”TENCON 2022 - 2022 IEEE Region 10 Conference (TENCON), 2022, pp. 1–6, doi: 10.1109/TENCON55691.2022.9978082（査読あり）.
2. 立川大雅，北須賀輝明，中西透，“近くにいる知人との相互認証ファイル共有アプリケーション，” 研究報告ユビキタスコンピューティングシステム（UBI), 2023 - UBI - 80 巻，37 号，pp. 1–7，2023（査読なし）.