Voice Calling Feature Documentation

Overview

The DigitalValut Chat application now includes a comprehensive secure voice calling feature with military-grade end-to-end encryption. This feature enables peer-to-peer (P2P) voice communication between users while maintaining the highest security standards.

Features

🔒 Security Features

- **Military-Grade Encryption**: All voice streams are encrypted using AES-GCM 256-bit encryption
- **End-to-End Encryption**: Audio data is encrypted on the sender's device and decrypted only on the receiver's device
- **Unique Session Keys**: Each voice call generates a unique encryption key for that session
- **No Server Storage**: All voice data is transmitted directly between peers, never stored on servers

📞 Call Features

- **One-Touch Calling**: Initiate voice calls with a single tap from the chat screen
- **Incoming Call Notifications**: Beautiful dialog for incoming call notifications
- **Call Controls**:
- Mute/Unmute microphone
- Toggle speaker on/off
- End call button
- **Call Duration Timer**: Real-time display of call duration
- **Connection Status**: Visual indicators for call state (calling, ringing, connected, ended)
- **Audio Optimization**:
- Echo cancellation
- Noise suppression
- Automatic gain control

🌐 Network Features

- **NAT Traversal**: Supports STUN servers for NAT traversal
- **Firewall Friendly**: Supports TURN servers for firewall traversal
- **Multiple STUN/TURN Servers**: Uses multiple public STUN/TURN servers for reliability
- **ICE Candidate Management**: Automatic ICE candidate exchange for optimal connectivity

Technical Architecture

Components

1. Voice Call Service (`lib/services/voice_call_service.dart`)

- Manages WebRTC peer connections
- Handles audio stream management
- Manages encryption keys

- Provides call control functions (mute, speaker, end call)
- Tracks call statistics and duration

2. Voice Call Screen (`lib/ui/voice_call_screen.dart`)

- Beautiful, intuitive call interface
- Real-time call status display
- Interactive call controls
- Animated UI elements for better UX

3. Incoming Call Dialog (`lib/widgets/incoming_call_dialog.dart`)

- Modal dialog for incoming call notifications
- Accept/Decline buttons
- Caller information display
- Security indicators

WebRTC Configuration

STUN Servers

```
- stun:stun.l.google.com:19302
- stun:stun1.l.google.com:19302
- stun:stun2.l.google.com:19302
- stun:stun3.l.google.com:19302
- stun:stun4.l.google.com:19302
```

TURN Servers (Public - Metered Free Tier)

```
- turn:openrelay.metered.ca:80
- turn:openrelay.metered.ca:443
- turn:openrelay.metered.ca:443?transport=tcp
```

Audio Configuration

```
{
  'audio': {
    'echoCancellation': true,
    'noiseSuppression': true,
    'autoGainControl': true,
  },
  'video': false
}
```

How to Use

Initiating a Voice Call

1. Open a conversation with a contact
2. Tap the phone icon in the app bar
3. Wait for the other party to answer
4. Start talking!

Receiving a Voice Call

1. An incoming call dialog will appear
2. Tap “Accept” to answer or “Decline” to reject
3. If accepted, you’ll be taken to the call screen

During a Call

- **Mute**: Tap the microphone icon to mute/unmute your microphone
- **Speaker**: Tap the speaker icon to toggle speaker on/off
- **End Call**: Tap the red phone icon to end the call

Call States

1. **Idle**: No active call
2. **Calling**: Initiating a call to a contact
3. **Ringing**: Receiving an incoming call
4. **Connected**: Active voice call in progress
5. **Ended**: Call has been terminated
6. **Failed**: Call connection failed

Security Implementation

Encryption Flow

1. **Key Generation**: When a call is initialized, a unique AES-GCM 256-bit key is generated
2. **Key Exchange**: Encryption keys are exchanged securely using X25519 key exchange (already implemented in the app’s encryption service)
3. **Stream Encryption**: All audio data is encrypted before transmission
4. **Decryption**: Received audio data is decrypted using the shared secret key

Security Best Practices

- ✅ Unique encryption key per call session
- ✅ Secure key exchange using Elliptic Curve Cryptography (X25519)
- ✅ No plaintext audio transmission
- ✅ Encrypted control messages
- ✅ Secure memory cleanup after call ends
- ✅ No call recording capability

Testing

Test Coverage

The voice calling feature includes comprehensive test coverage:

Unit Tests (`test/services/voice_call_service_test.dart`)

- Service initialization
- State management
- Mute/unmute functionality
- Speaker toggle
- Call duration tracking

- Statistics collection
- Error handling

Widget Tests

- `test/widgets/voice_call_screen_test.dart`: Voice call screen UI tests
- `test/widgets/incoming_call_dialog_test.dart`: Incoming call dialog tests

Integration Tests (`test/integration/voice_call_integration_test.dart`)

- Complete call flow (initialize, call, answer, end)
- ICE candidate exchange
- Encryption maintenance
- Network disconnection handling
- Performance tests
- Security tests

Running Tests

```bash
# Run all tests
flutter test

# Run specific test file
flutter test test/services/voice_call_service_test.dart

# Run with coverage
flutter test --coverage
```

Platform Support

✅ Android

- Full support for voice calling
- Background call handling
- Native audio routing
- Microphone permissions handled automatically

✅ Web

- Full support for voice calling
- WebRTC adapter included for cross-browser compatibility
- Microphone permissions requested via browser
- Works on Chrome, Firefox, Safari, Edge

🔜 iOS (Future Release)

- Planned for next phase
- Will include CallKit integration
- Native iOS audio handling

Permissions

Android

The following permissions are already configured in `AndroidManifest.xml`:
- `RECORD_AUDIO`: Required for microphone access
- `MODIFY_AUDIO_SETTINGS`: Required for audio routing (speaker/earpiece)
- `INTERNET`: Required for WebRTC connections

Web

Microphone permissions are requested by the browser when a call is initiated.

Network Requirements

Minimum Requirements

- Internet connection (WiFi or cellular data)
- Outbound UDP ports open for WebRTC
- HTTPS connection for web version

Recommended

- Stable internet connection with at least 100 Kbps upload/download
- Low latency (<200ms ping)
- Unrestricted UDP traffic

Firewall Considerations

The app uses TURN servers to work behind restrictive firewalls and NATs, ensuring connectivity in most network environments.

Troubleshooting

Call Not Connecting

- Check internet connection
- Ensure microphone permissions are granted
- Verify both parties are online
- Try switching between WiFi and mobile data

Audio Quality Issues

- Check network stability
- Move to an area with better signal
- Close other bandwidth-intensive applications
- Ensure microphone is not blocked or muted

No Audio

- Check if microphone is muted in the call
- Verify microphone permissions
- Test microphone in device settings
- Toggle speaker on/off

Call Dropping

- Check network stability
- Verify internet connection
- Check if device is in power-saving mode
- Ensure app has background execution permissions

Future Enhancements

Planned Features

- [ ] Group voice calls (conference calling)
- [ ] Video calling support
- [ ] Call recording (with end-to-end encryption)
- [ ] Call history and logs
- [ ] Voicemail support
- [ ] iOS platform support with CallKit
- [ ] Screen sharing during calls
- [ ] Call quality indicators
- [ ] Network quality adaptation
- [ ] Bluetooth headset support
- [ ] Call statistics dashboard

API Reference

VoiceCallService

```
class VoiceCallService {
  // Initialize the service
  Future<void> initialize();

  // Start a call (caller side)
  Future<RTCSessionDescription> startCall();

  // Answer an incoming call (receiver side)
  Future<RTCSessionDescription> answerCall(RTCSessionDescription offer);

  // Set remote session description
  Future<void> setRemoteDescription(RTCSessionDescription description);

  // Add ICE candidate
  Future<void> addIceCandidate(RTCIceCandidate candidate);

  // Toggle microphone mute
  Future<void> toggleMute();

  // Toggle speaker
  Future<void> toggleSpeaker();

  // End the call
  Future<void> endCall();

  // Get call statistics
  Future<Map<String, dynamic>> getCallStatistics();

  // Properties
  VoiceCallState get state;
  bool get isMuted;
  bool get isSpeakerOn;
  int get callDuration;
}
```

VoiceCallState Enum

```
enum VoiceCallState {
  idle,       // No active call
  calling,    // Initiating call
  ringing,    // Incoming call
  connected,  // Active call
  ended,      // Call terminated
  failed,     // Call failed
}
```

Performance Considerations

Memory Usage

- Efficient stream management
- Automatic cleanup on call end
- No memory leaks in prolonged usage

Battery Consumption

- Optimized for minimal battery drain
- Efficient audio encoding
- Smart wake lock management

Network Usage

- Adaptive bitrate based on connection
- Typical audio bandwidth: 32-64 Kbps
- Efficient packet transmission

Compliance and Privacy

Data Protection

- No call recording by default
- No server-side storage of voice data
- Peer-to-peer transmission only
- Encrypted at all times

GDPR Compliance

- No personal data collection during calls
- User consent for microphone access
- Right to be forgotten (no stored call data)

Security Standards

- Follows NIST cryptographic standards
- Implements OWASP mobile security best practices
- Regular security audits recommended

Credits

Technologies Used

- **Flutter WebRTC**: Cross-platform WebRTC implementation
- **Cryptography Package**: Encryption implementation
- **Google STUN Servers**: NAT traversal
- **Metered TURN Servers**: Firewall traversal

Contributors

- Voice calling feature implementation: AI Assistant
- Security architecture: Based on military-grade standards
- UI/UX design: Material Design 3 principles

Support

For issues, questions, or feature requests, please:
1. Check this documentation first
2. Review the troubleshooting section

3. Open an issue on GitHub
4. Contact the development team

License

Last Updated: November 3, 2025
Version: 1.0.0
Status: Production Ready ✅