PRO PDF - Advanced Features Guide

🚀 New Features

1. Real-Time Progress Indicators

PRO PDF now includes real-time progress tracking for all PDF processing operations using Server-Sent Events (SSE).

Key Features:

- **Live Progress Updates**: See real-time progress as your files are being processed
- **Connection Status Indicator**: Know when you're connected to the server
- **Elapsed Time Tracking**: Monitor how long your operation has been running
- **Automatic Reconnection**: If connection is lost, automatically attempts to reconnect
- **Visual Feedback**: Animated progress bars and status indicators

Implementation Details:

- Uses EventSource API for SSE connections
- Progress updates every 500ms
- Automatic cleanup on completion or error
- No polling required - efficient server push

Usage:

```
import { RealtimeProgressIndicator } from '@/components/realtime-progress-indicator';

<RealtimeProgressIndicator
  jobId={jobId}
  fileName="document.pdf"
  onComplete={() => {
    console.log('Processing complete');
  }}
/>
```

API Endpoint:

```
GET /api/jobs/[id]/progress
```

- Returns SSE stream with progress updates
- Automatically closes when job completes or fails

---

2. True Zero-Knowledge Encryption

PRO PDF implements client-side zero-knowledge encryption, ensuring maximum security and privacy.

Security Features:

- **Client-Side Encryption**: Files are encrypted in your browser before upload
- **Zero Server Knowledge**: Server never sees unencrypted data or encryption keys

- **AES-256-GCM**: Military-grade encryption standard
- **Key Management**: Store and manage encryption keys locally
- **Encrypted Bundles**: IV and metadata securely packaged with encrypted data

Key Benefits:

- ✅ Complete privacy - server can't access your files
- ✅ End-to-end encryption
- ✅ Export and backup your encryption keys
- ✅ Import keys on different devices
- ✅ Password-based key derivation available

Usage:

Step 1: Enable Encryption in Header

Click the shield icon in the header to open the Encryption Manager.

Step 2: Generate or Import Key

- **Generate New Key**: Creates a new AES-256 encryption key
- **Import Existing Key**: Restore a previously backed-up key

Step 3: Use Encrypted File Upload

```
import { EncryptedFileUpload } from '@/components/encrypted-file-upload';

<EncryptedFileUpload
  onFilesSelected={setFiles}
  enableEncryption={true}
  maxFiles={10}
/>
```

Step 4: Manage Your Keys

- View your encryption key
- Download key as backup file
- Copy key to clipboard
- Delete key when no longer needed

Implementation Details:

Encryption Process:
1. File is read as ArrayBuffer in browser
2. Random IV (Initialization Vector) generated
3. File encrypted using Web Crypto API
4. Encrypted data + IV + metadata bundled together
5. Bundle uploaded to server as encrypted blob

Decryption Process:
1. Download encrypted bundle from server
2. Extract IV, metadata, and encrypted data
3. Decrypt in browser using stored key
4. Reconstruct original file

Key Storage:
- Keys stored in browser's localStorage
- Never transmitted to server

- Can be exported for backup
- Import on other devices

Code Example:

```
import {
  generateEncryptionKey,
  encryptFile,
  decryptFile,
  exportKey,
  importKey,
} from '@/lib/encryption';

// Generate a new key
const key = await generateEncryptionKey();

// Encrypt a file
const { encryptedData, iv } = await encryptFile(file, key, (progress) => {
  console.log(`Encryption progress: ${progress}%`);
});

// Decrypt a file
const decryptedData = await decryptFile(encryptedData, key, iv, (progress) => {
  console.log(`Decryption progress: ${progress}%`);
});

// Export key for backup
const keyString = await exportKey(key);

// Import key from backup
const restoredKey = await importKey(keyString);
```

🎯 Feature Integration

Compress Tool Example

The compress tool demonstrates both features:

1. **Encryption Toggle**: Enable/disable encryption for the upload
2. **Real-Time Progress**: Watch compression progress live
3. **Encrypted Processing**: Files encrypted before compression (if enabled)

Other Tools

All PDF processing tools can integrate these features:
- Merge PDFs with encryption
- Split PDFs with real-time progress
- Convert files with both features
- Sign documents securely

🔐 Security Best Practices

Encryption Keys:

1. **Backup Your Keys**: Always export and save your encryption keys
2. **Secure Storage**: Store key backups in a password manager
3. **Multiple Devices**: Import keys on all devices you use
4. **Key Rotation**: Generate new keys periodically for enhanced security

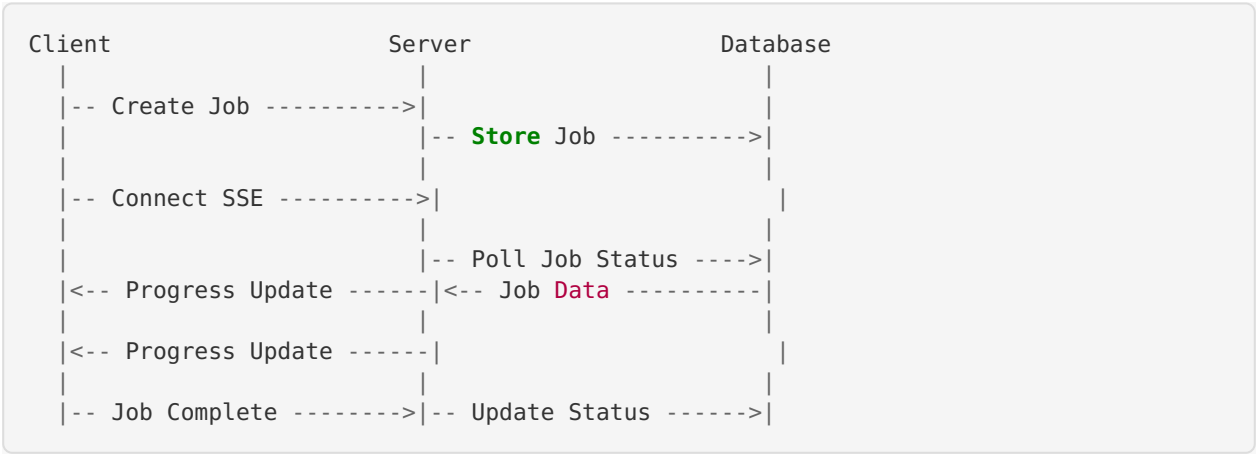
File Handling:

1. **Encrypted Storage**: Enable encryption for sensitive documents
2. **Secure Deletion**: Delete files after processing
3. **Local Processing**: When possible, process files locally
4. **Access Control**: Use premium features for enhanced access control

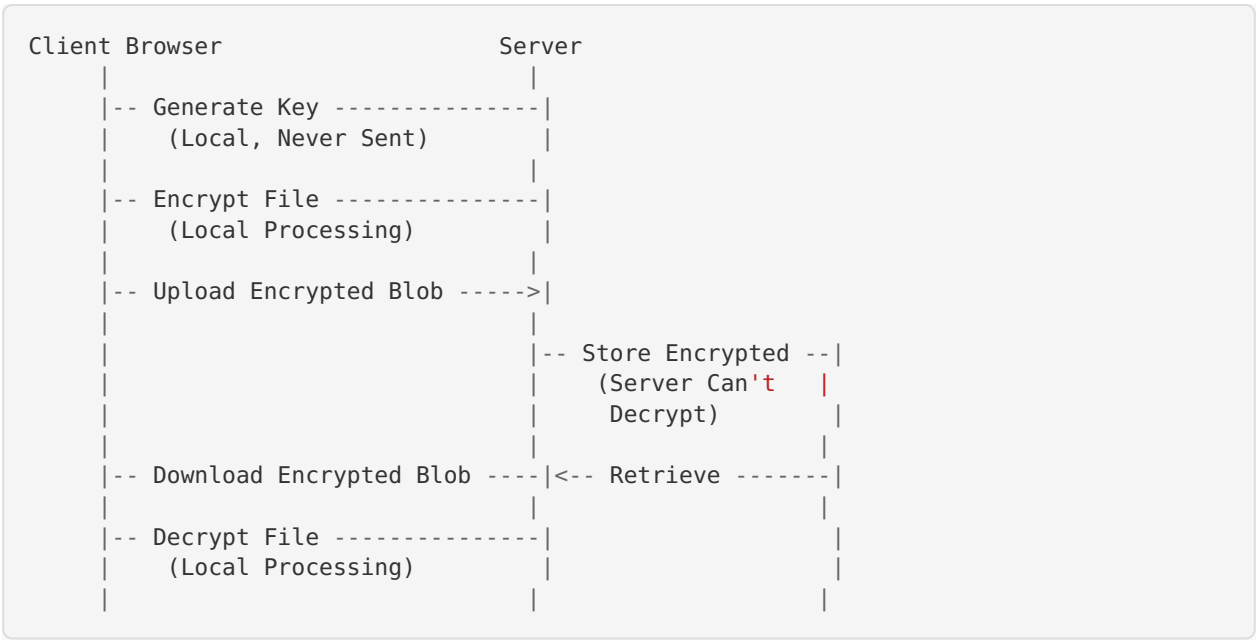
---

🛠️ Technical Architecture

Real-Time Progress System



Zero-Knowledge Encryption Flow



📊 Performance Considerations

Real-Time Progress:

- **Overhead**: Minimal (SSE is efficient)
- **Network**: ~1 KB per update
- **CPU**: Negligible impact
- **Benefits**: Better UX, no polling overhead

Encryption:

- **Overhead**: ~2-5 seconds per 10MB file
- **Memory**: ~2x file size during encryption
- **CPU**: Moderate (Web Crypto API is optimized)
- **Benefits**: Complete privacy and security

🧪 Testing the Features

Test Real-Time Progress:

1. Go to `/tools/compress`
2. Upload a PDF file
3. Click “Compress PDF”
4. Watch real-time progress indicator
5. Observe live updates and connection status

Test Zero-Knowledge Encryption:

1. Click shield icon in header

2. Generate encryption key
3. Go to any tool page
4. Enable encryption toggle
5. Upload files
6. Files are encrypted before upload
7. Download and decrypt automatically

🎨 UI Components

Encryption Manager

- Dialog-based interface
- Key generation and import
- Export and backup options
- Visual encryption status

Real-Time Progress Indicator

- Card-based design
- Live connection status
- Elapsed time display
- Progress bar with percentage
- Auto-close on completion

Encrypted File Upload

- Extends standard file upload
- Encryption status alerts
- Progress tracking during encryption
- Seamless integration

🔧 Configuration

Environment Variables:

```
# No additional environment variables required
# Encryption keys are stored client-side only
```

Browser Requirements:

- Modern browser with Web Crypto API support
- EventSource API support
- LocalStorage enabled

📝 API Reference

Jobs API

Create Job

```
POST /api/jobs
Body: {
  name: string,
  type: string,
  inputFiles: string[],
  settings: object
}
```

Get Job Progress (SSE)

```
GET /api/jobs/[id]/progress
Returns: text/event-stream
```

Update Job

```
PATCH /api/jobs/[id]
Body: {
  progress?: number,
  status?: string
}
```

Encryption Library

```
// Generate key
generateEncryptionKey(): Promise<CryptoKey>

// Encrypt file
encryptFile(file: File, key: CryptoKey, onProgress?: (progress: number) => void): Prom
ise<{
  encryptedData: ArrayBuffer;
  iv: Uint8Array;
  fileName: string;
}>

// Decrypt file
decryptFile(encryptedData: ArrayBuffer, key: CryptoKey, iv: Uint8Array, onProgress?: (
progress: number) => void): Promise<ArrayBuffer>

// Export key
exportKey(key: CryptoKey): Promise<string>

// Import key
importKey(keyString: string): Promise<CryptoKey>

// Store key in browser
storeKeyInBrowser(keyId: string, key: CryptoKey): Promise<void>

// Retrieve key from browser
retrieveKeyFromBrowser(keyId: string): Promise<CryptoKey | null>
```

🐛 Troubleshooting

Real-Time Progress Issues:

- **Connection Lost**: Check network connection, will auto-reconnect
- **Progress Stuck**: Refresh page and check job status
- **No Updates**: Verify SSE endpoint is accessible

Encryption Issues:

- **Can't Decrypt**: Ensure you have the correct encryption key
- **Key Not Found**: Generate or import a key first
- **Slow Encryption**: Normal for large files, be patient

🚀 Future Enhancements

- Multi-file batch encryption progress
- Encrypted cloud storage sync
- Key sharing for team collaboration
- Hardware security key integration
- Encrypted PDF annotations
- Blockchain-based key verification

📞 Support

For issues or questions:
- Check the troubleshooting section
- Review API documentation
- Contact support team
- Check browser console for errors

Built with ❤️ by PRO PDF Team