

# Encapsulated Upgradable Components

Eric Allen  
Victor Luchangco  
Sam Tobin-Hochstadt

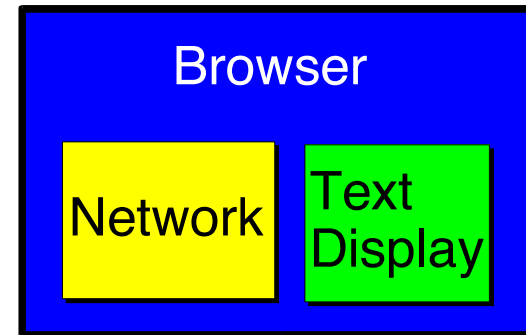
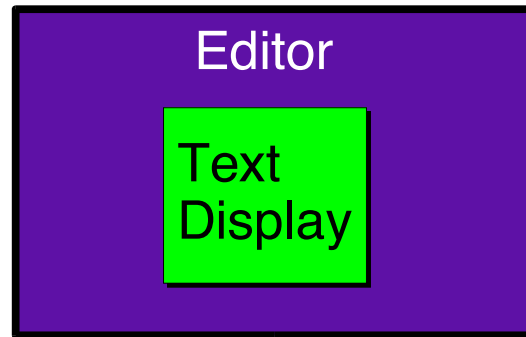
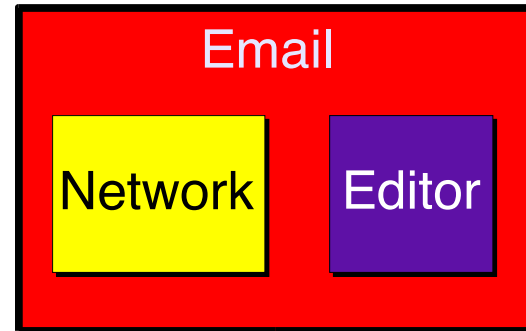
Sun Microsystems, Inc.

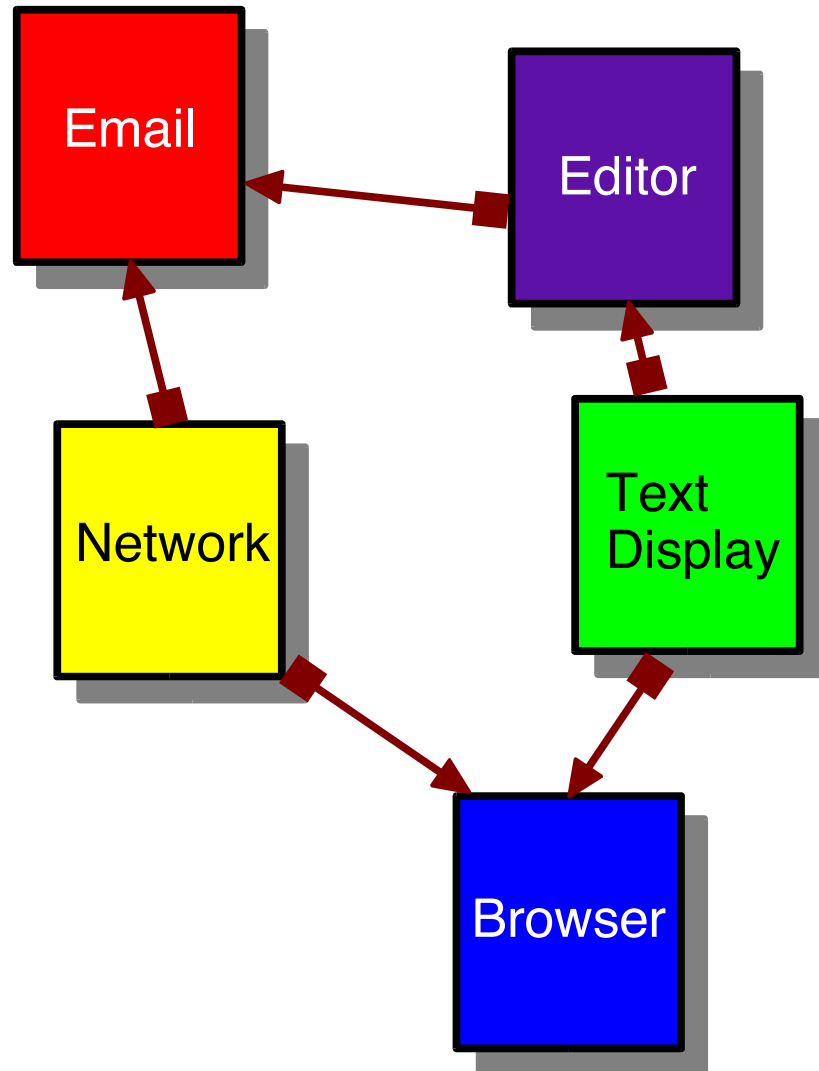


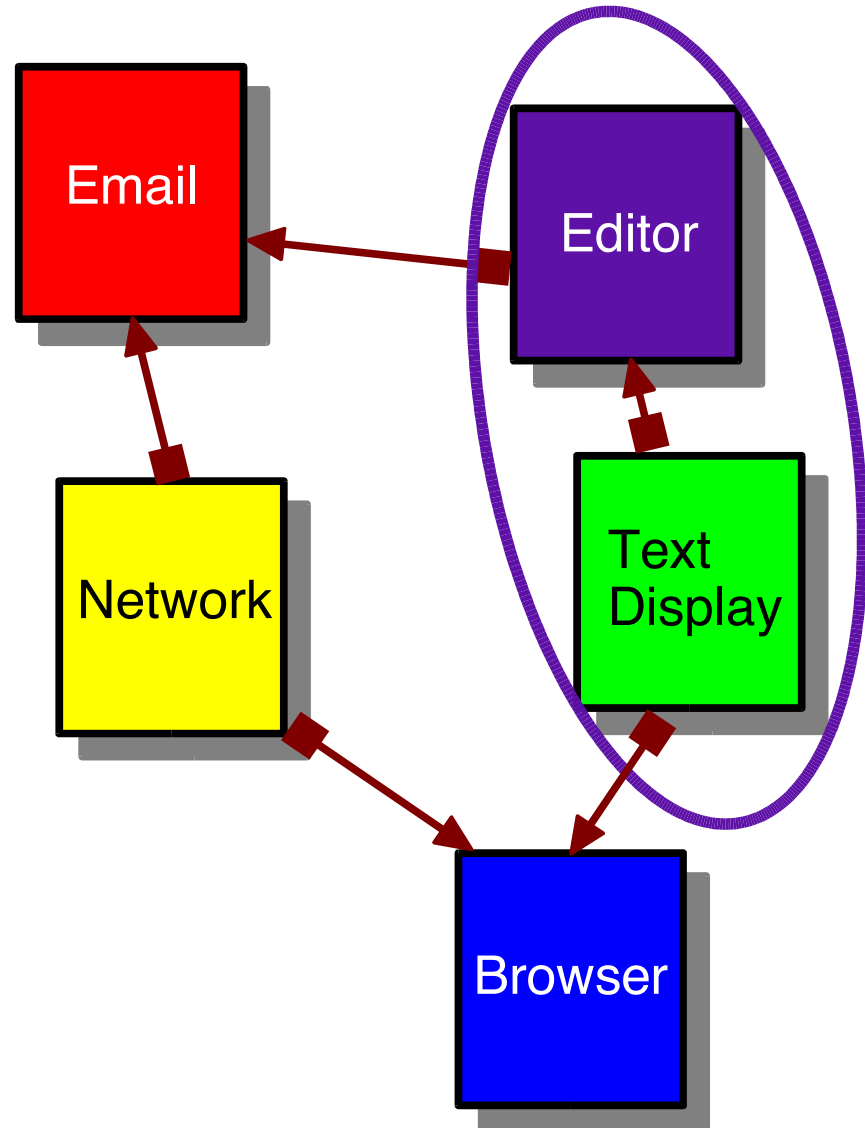
Email

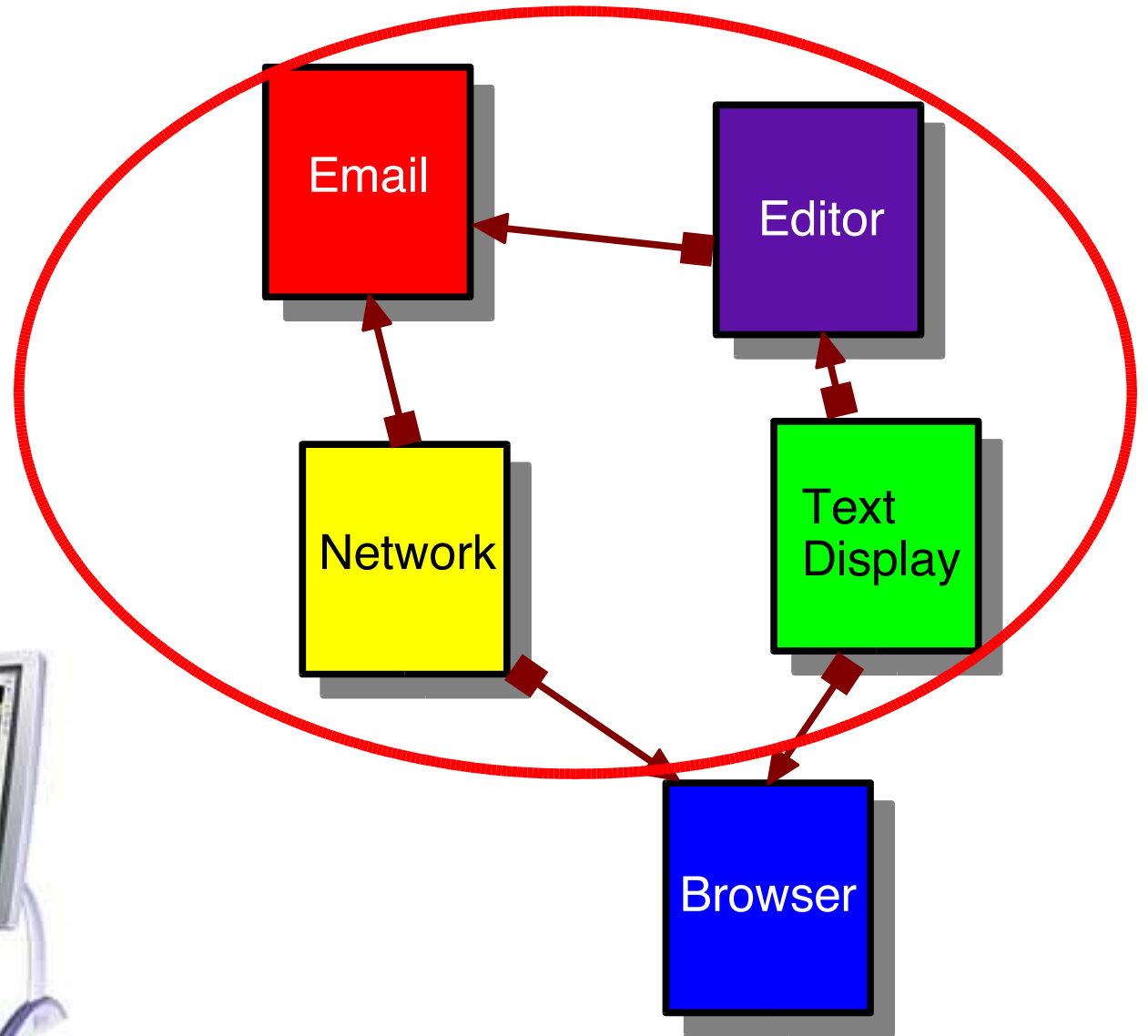
Editor

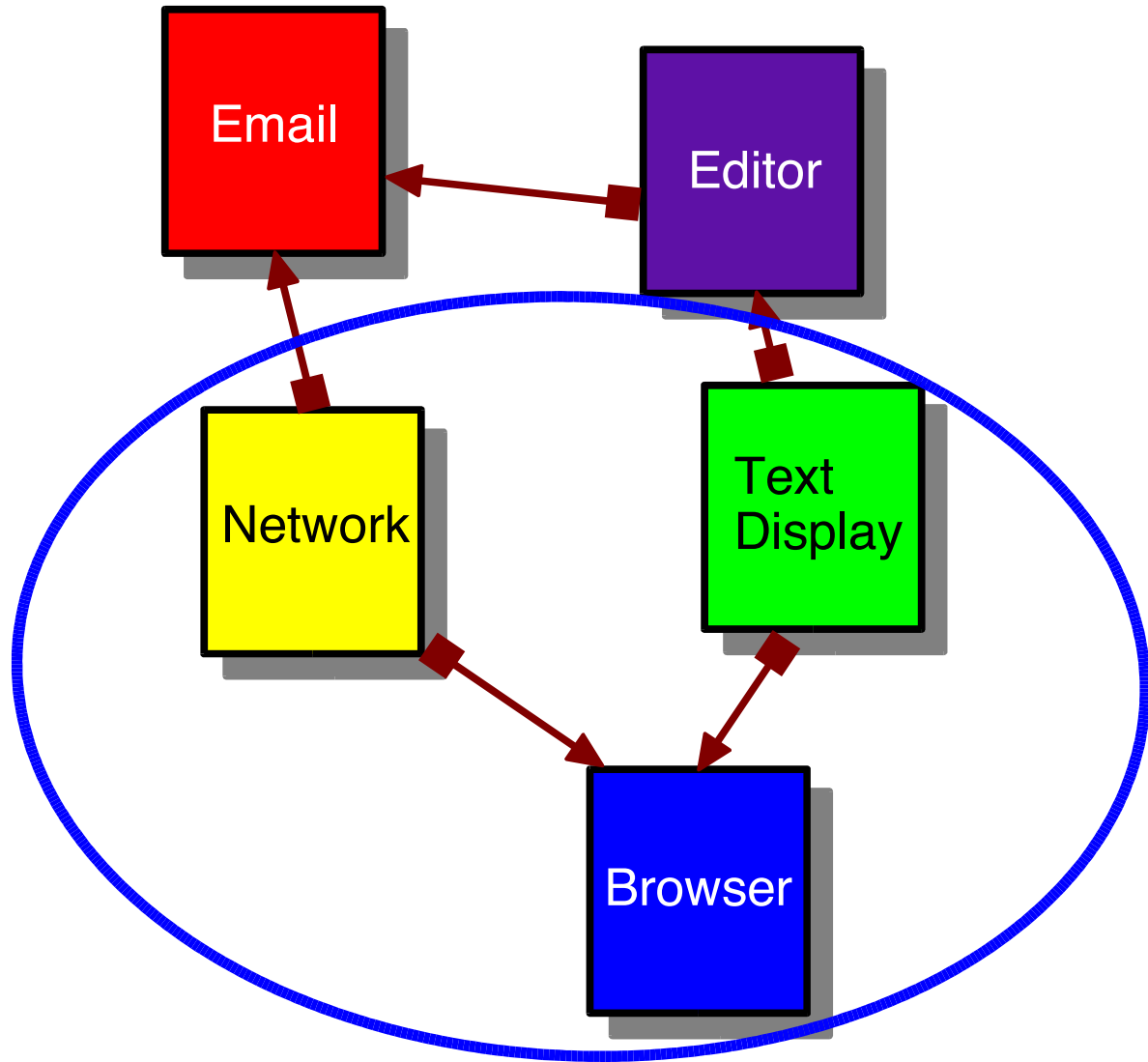
Browser

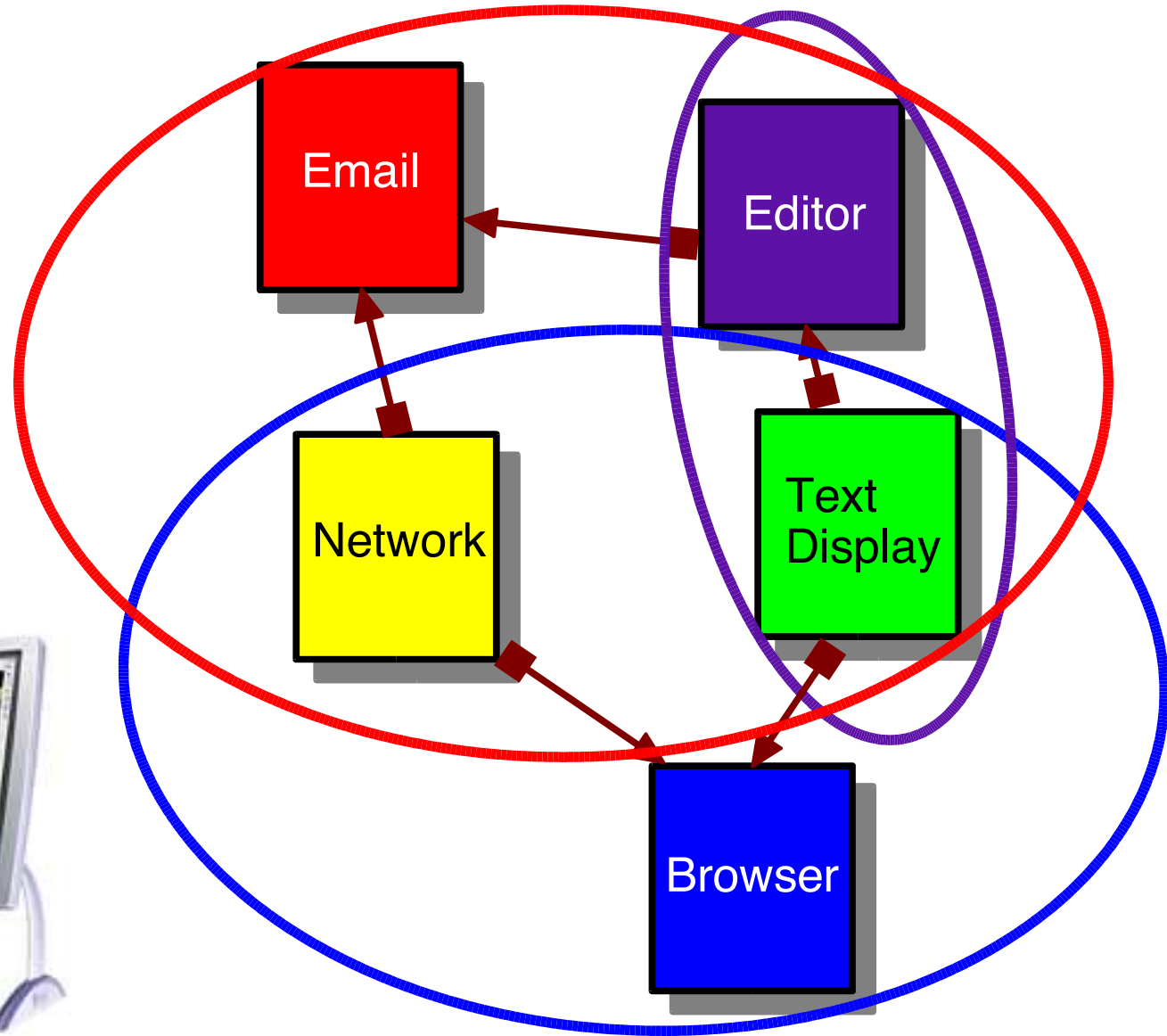












# Linking during software development

- Breaking up development tasks
- Reusing common code
- Switching libraries
- Testing programs

# Linking during software development

- Breaking up development tasks
- Reusing common code
- Switching libraries
- Testing programs

Solved with modules

```
structure Browser =  
  ...
```

```
structure Email =  
  ...
```

```
structure Editor =  
  ...
```

```
structure Network =  
  ...
```

```
structure TextDisplay =  
  ...
```

```
structure Email =  
  ...
```

```
structure Network =  
  ...
```

```
structure Editor =  
  ...
```

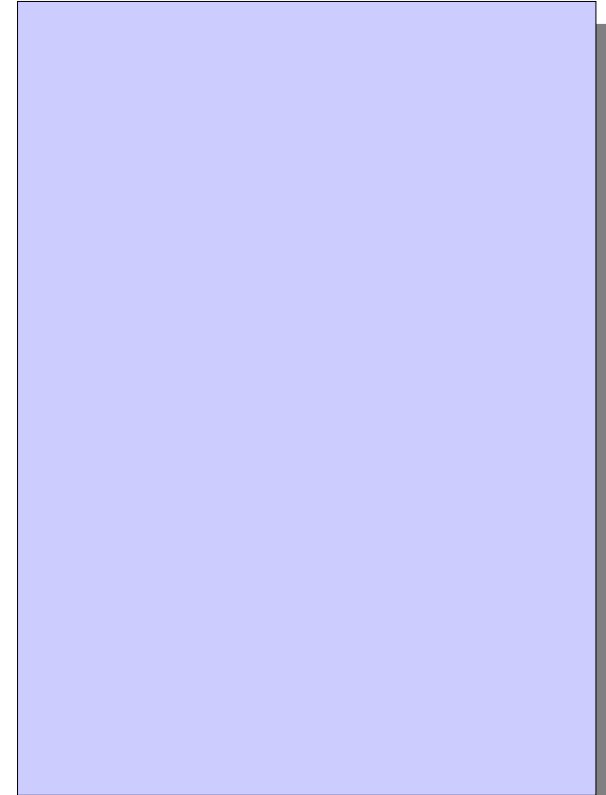
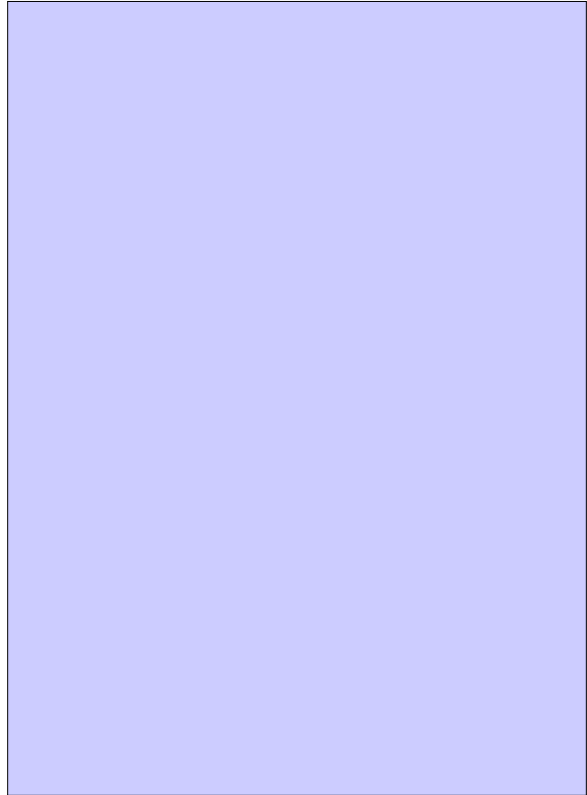
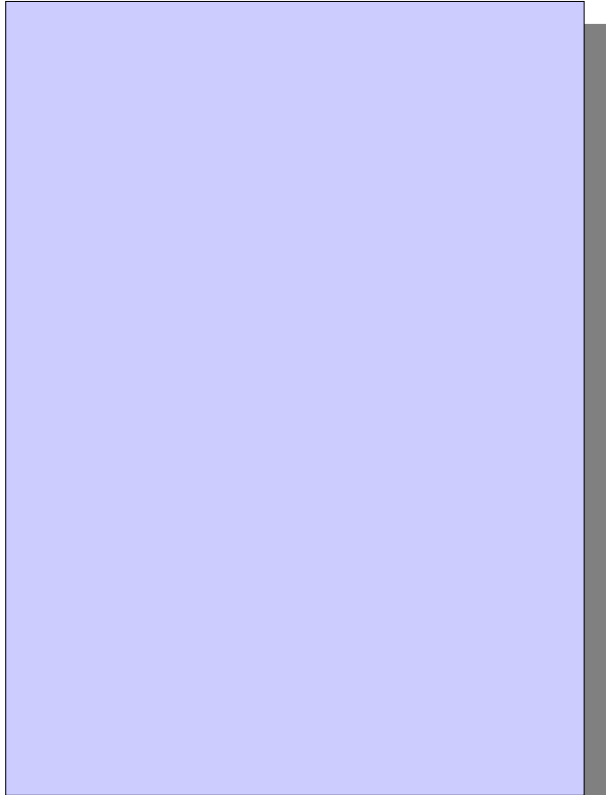
```
structure Browser =  
  ...
```

```
structure Network =  
  ...
```

```
structure  
  TextDisplay = ...
```

```
structure Editor =  
  ...
```

```
structure  
  TextDisplay = ...
```



# Module systems

- ML modules (Milner, Harper, MacQueen, Tofte)
- MzScheme units (Felleisen, Flatt)
- Jiazzi (Hsieh, Flatt, McDirmid)
- MJ (Bacon, Grove, Murthy, Corwin)
- Etc.

# Encapsulated Upgradable Components

- The stability of static linking
- The sharing and upgradability of dynamic linking

# Desired properties

- Installation never blocked by existing components
- Execution without signaling a component error
- Upgrade without affecting other applications
- No unnecessary copies

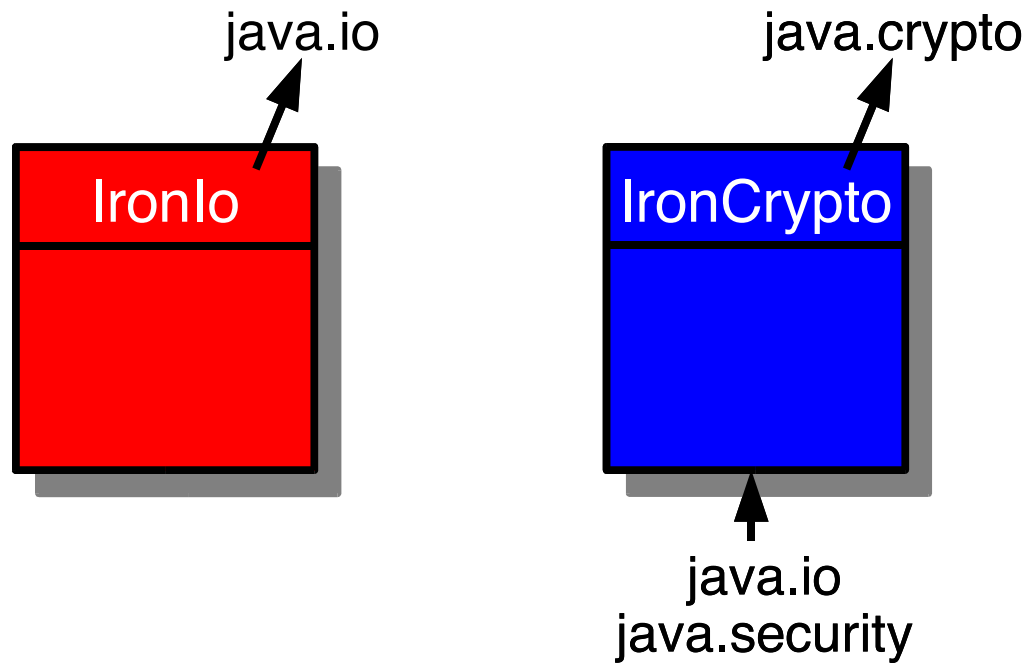
# Components

- Components are immutable
- **Simple components** are units of compilation
  - Typically the size of small Java packages
- **Compound components** are produced by combining components
  - Through linking
  - Through upgrade
- Components import and export **apis**

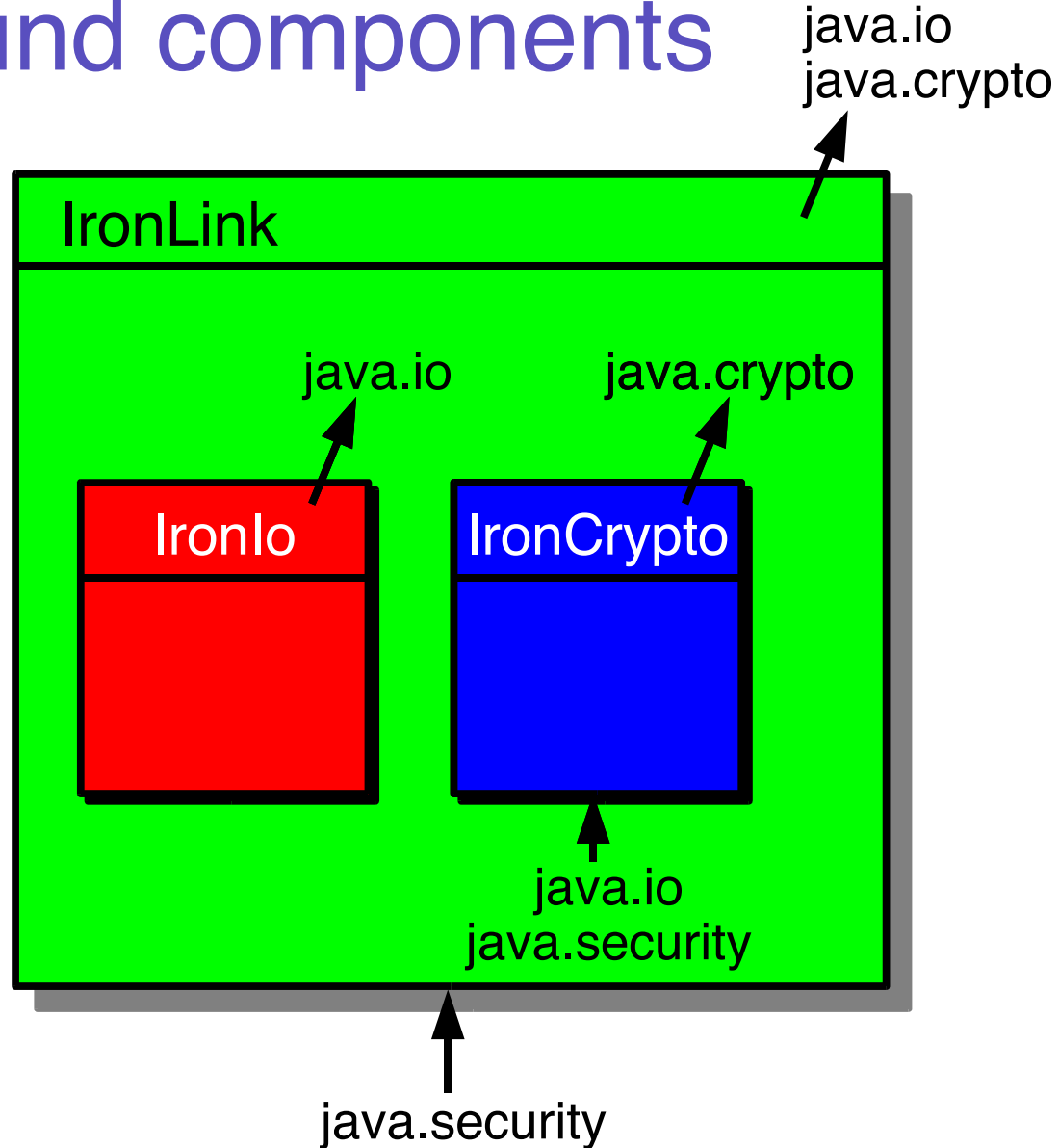
# Apis

- **Apis** are the “interfaces” of components
- Apis consist only of **declarations**, not definitions
- An api **imports** other apis it uses
- Each api in the world has a distinct name

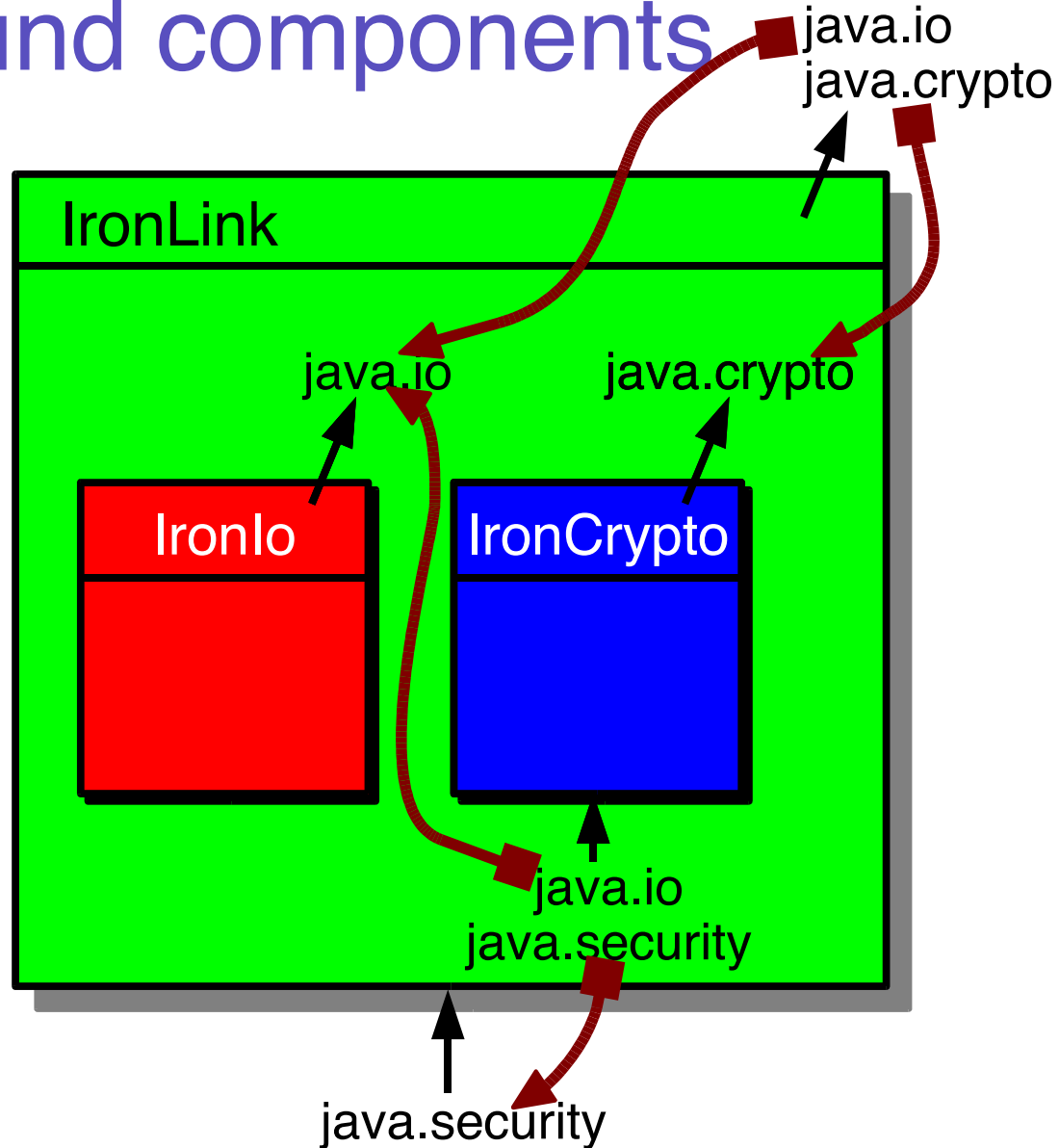
# Simple components



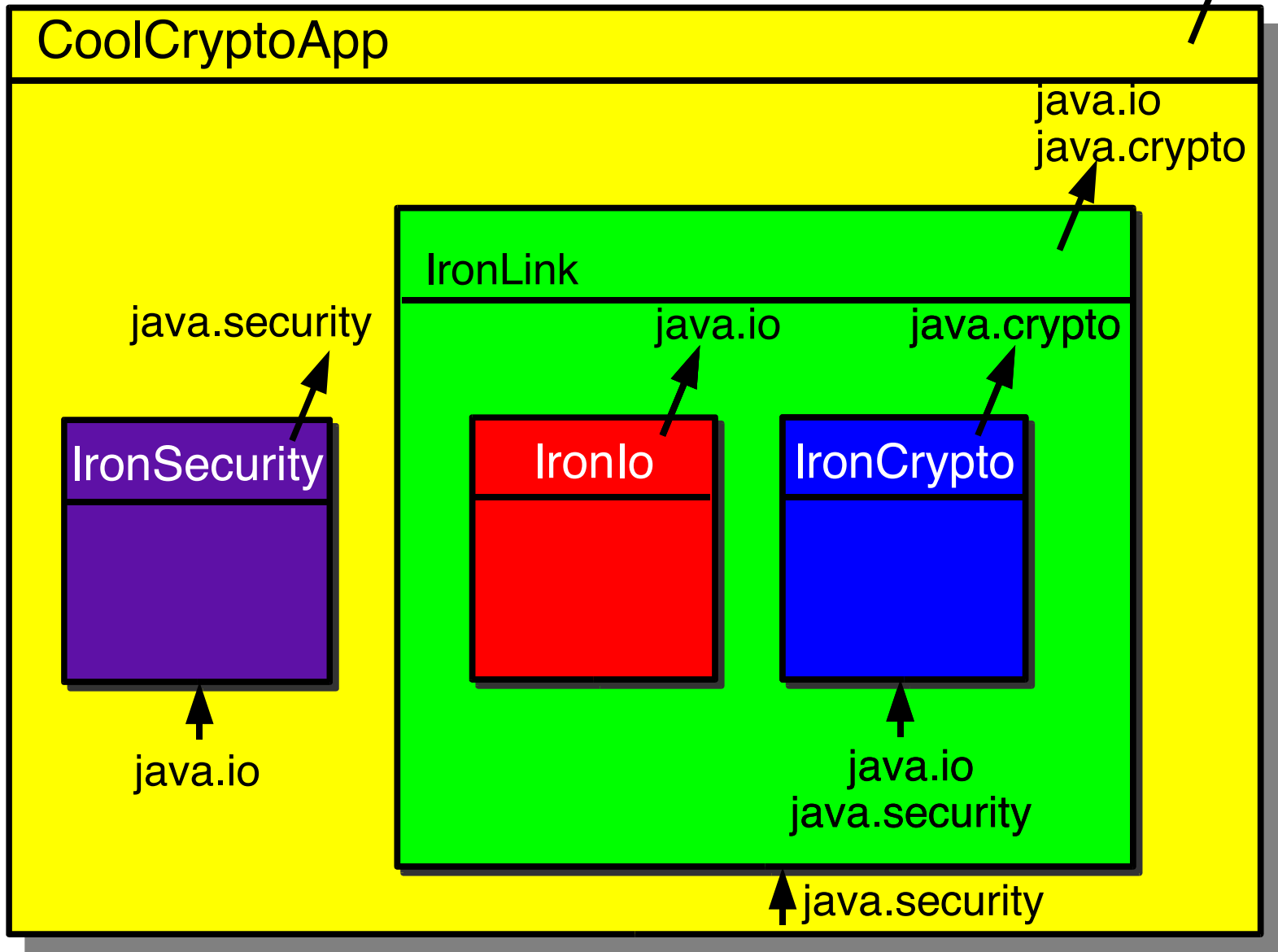
# Compound components



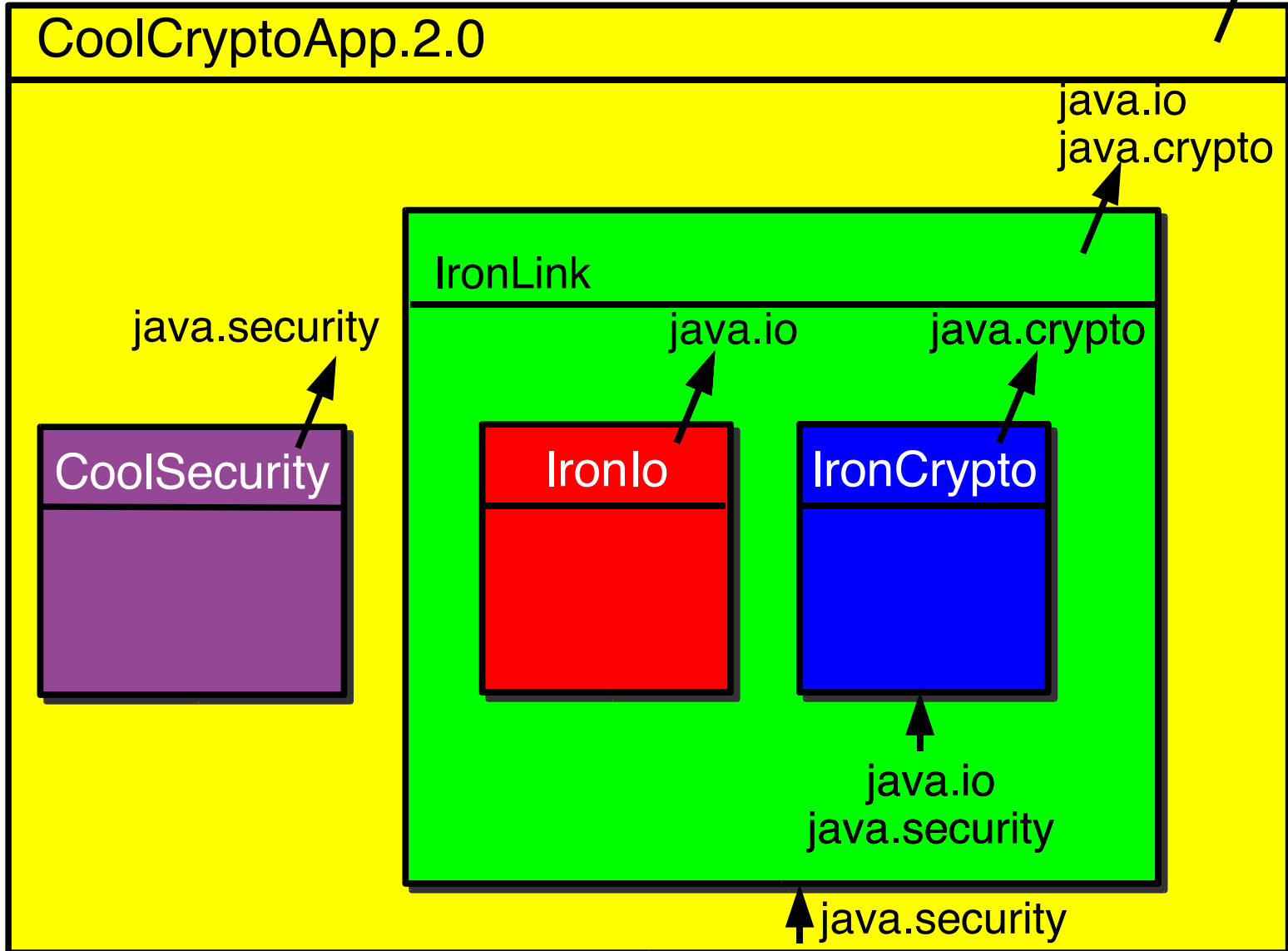
# Compound components



java.crypto, java.security

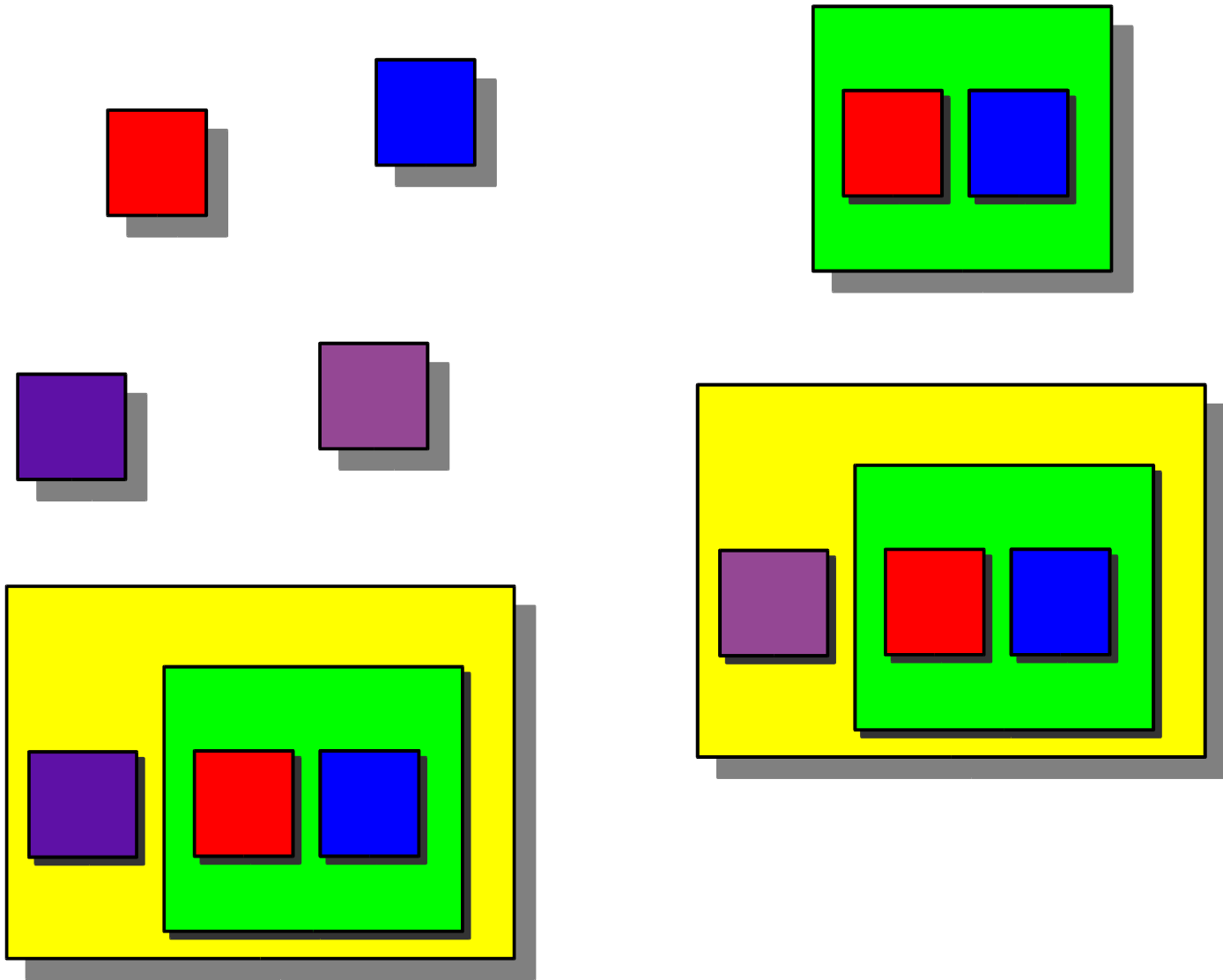


java.crypto, java.security



# Api upgradable

- Upgrade restrictions are specified via exporting the special **upgradable** api
- Imports a special **components** api
- Only api **upgradable** can be exported by multiple constituents



What about sharing?

# Fortresses

- Components are not manipulated directly; they are stored in **fortresses**
- Fortresses are persistent databases mapping names to components and apis
- Typically, a single machine includes a single fortress

# Efficient implementation

- Because components are immutable, they can be shared at will
- Components not directly reachable can be referred to by other compound components
- Reclamation of unused components can be handled via conventional garbage collection

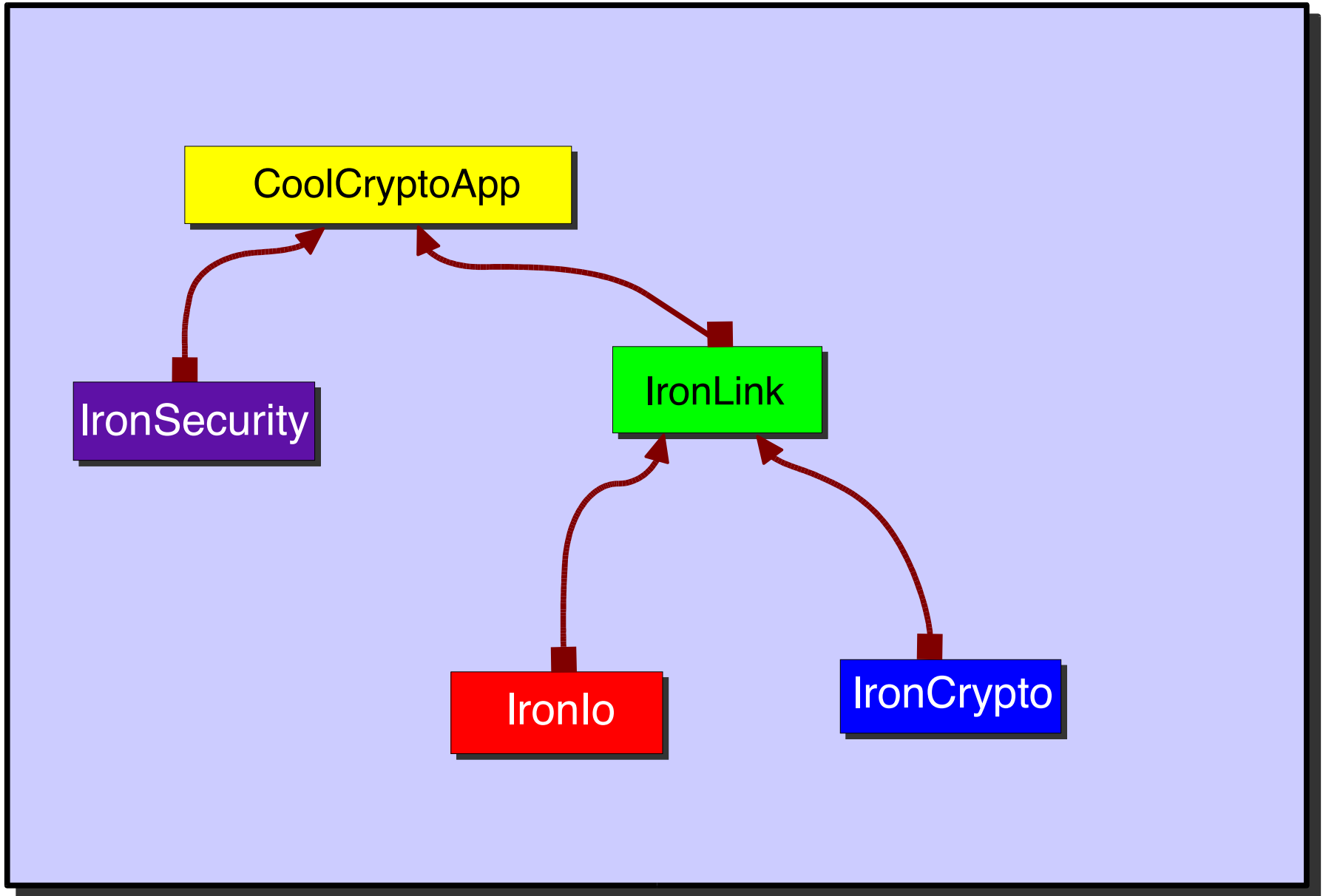
CoolCryptoApp

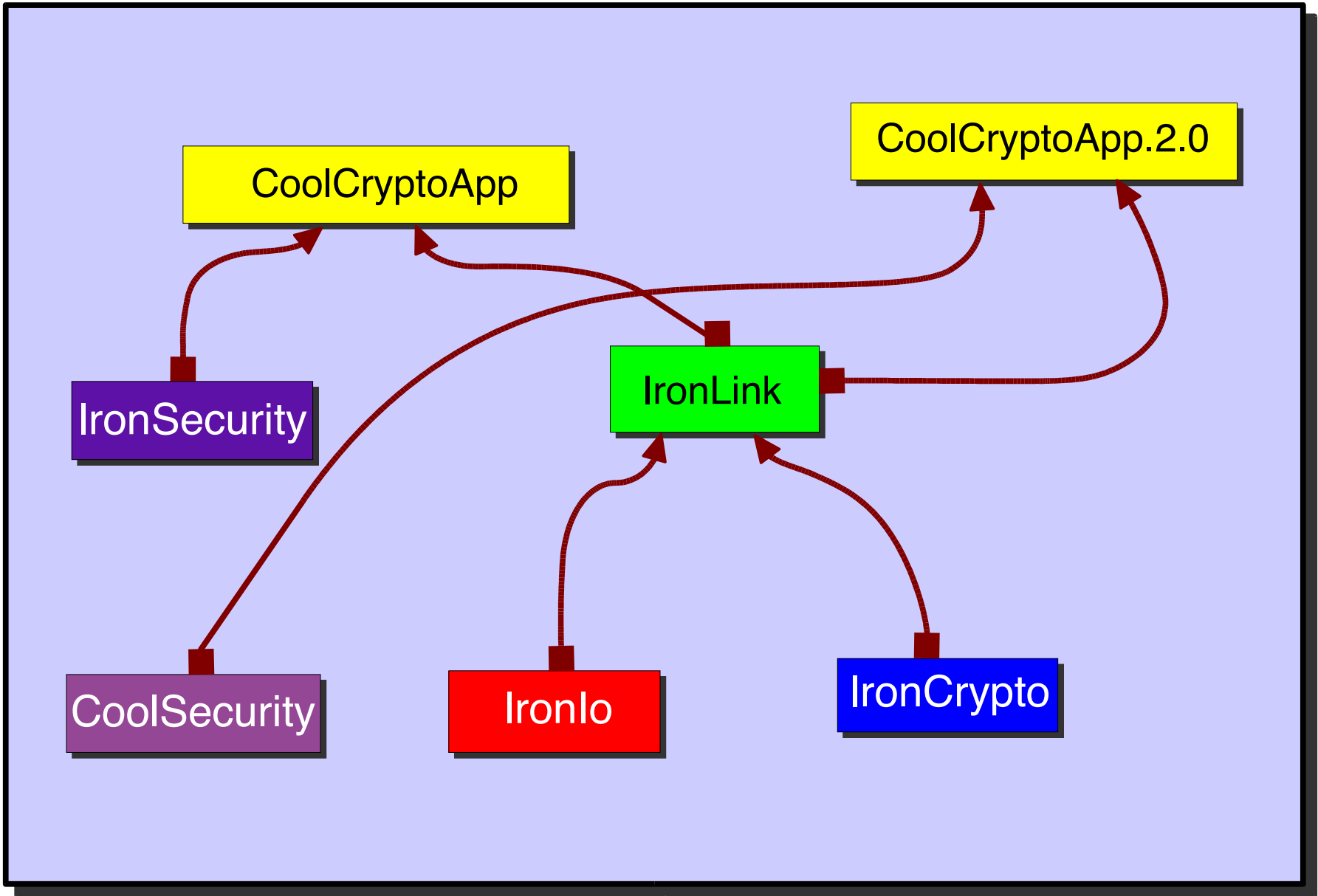
IronSecurity

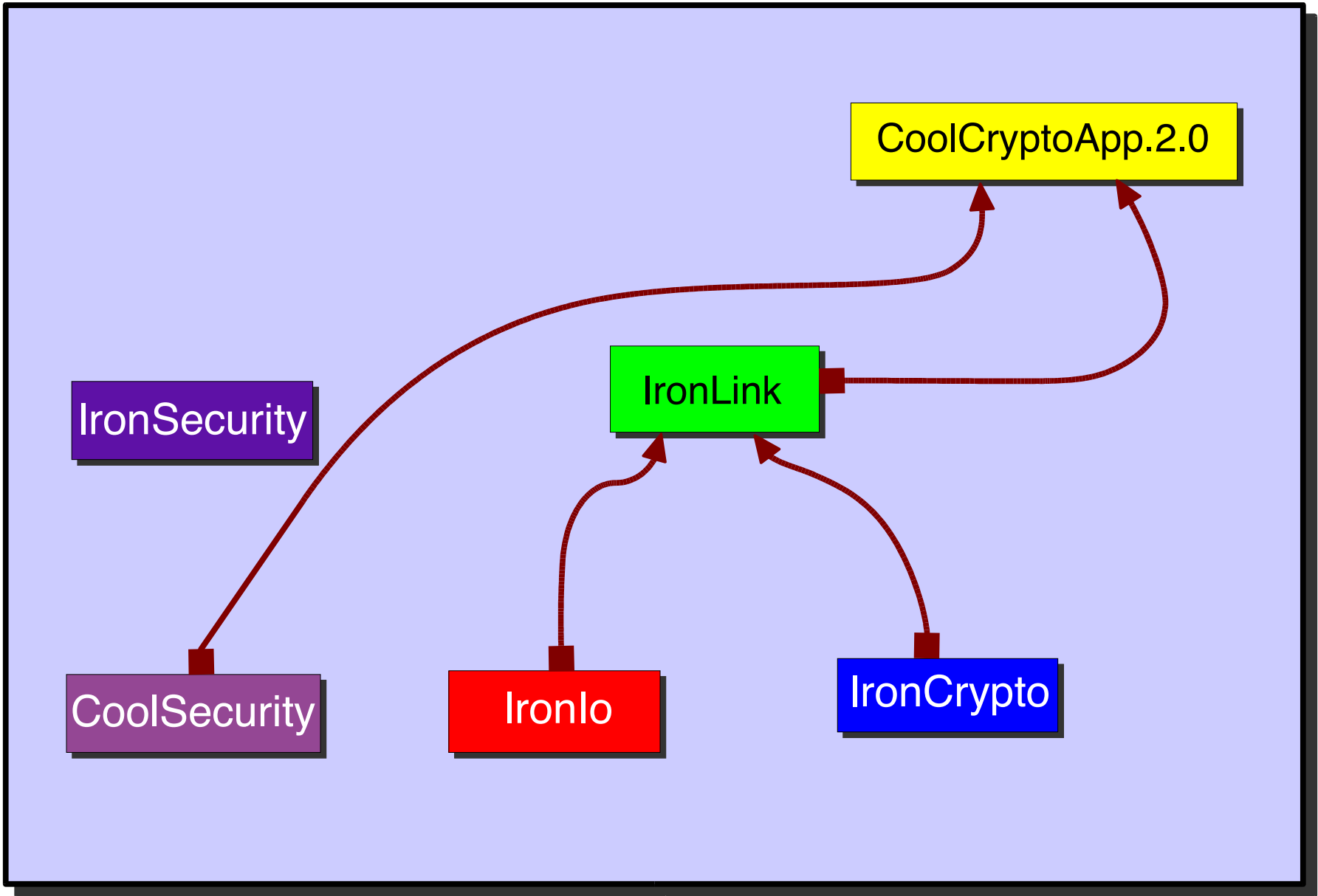
IronLink

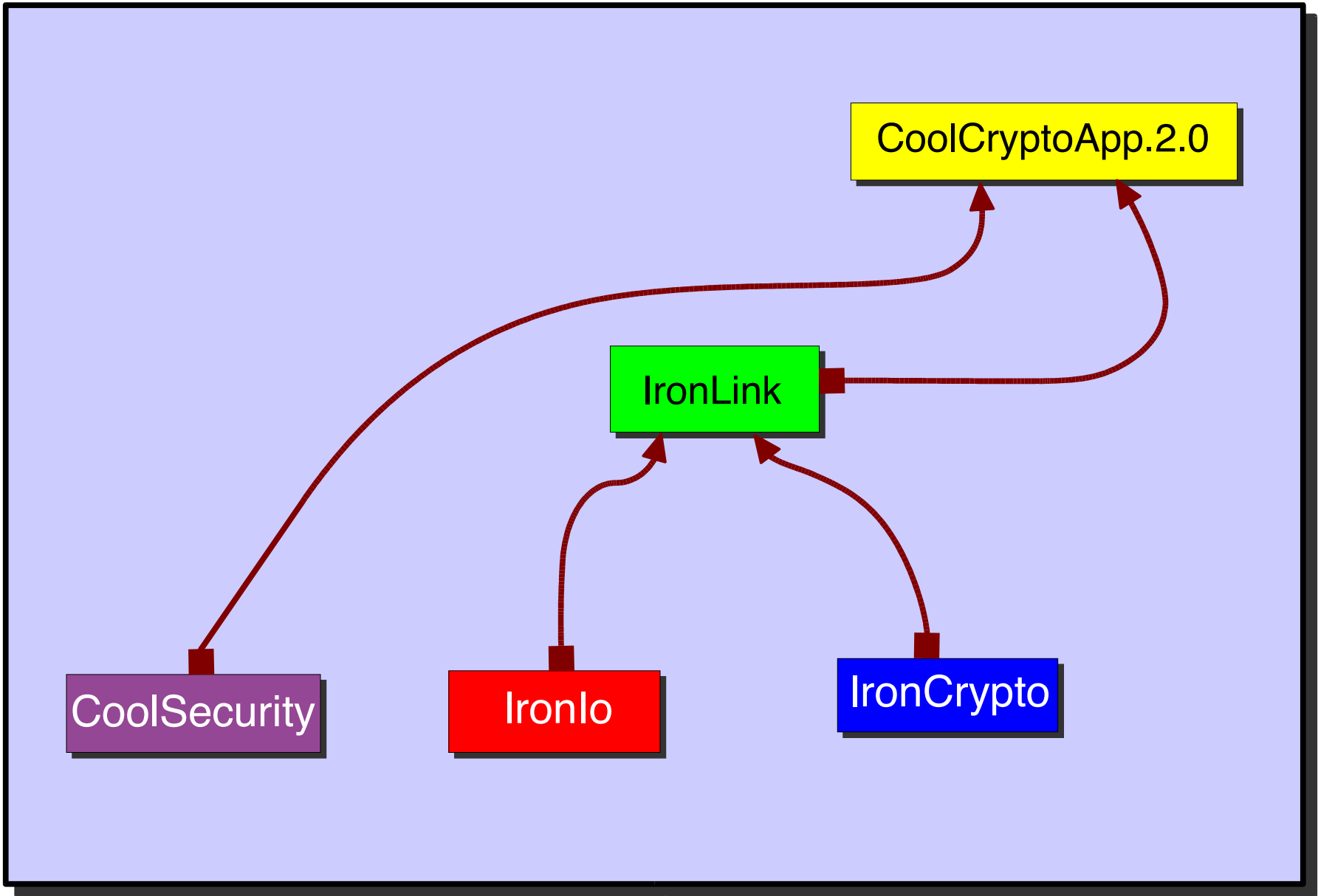
Ironlo

IronCrypto



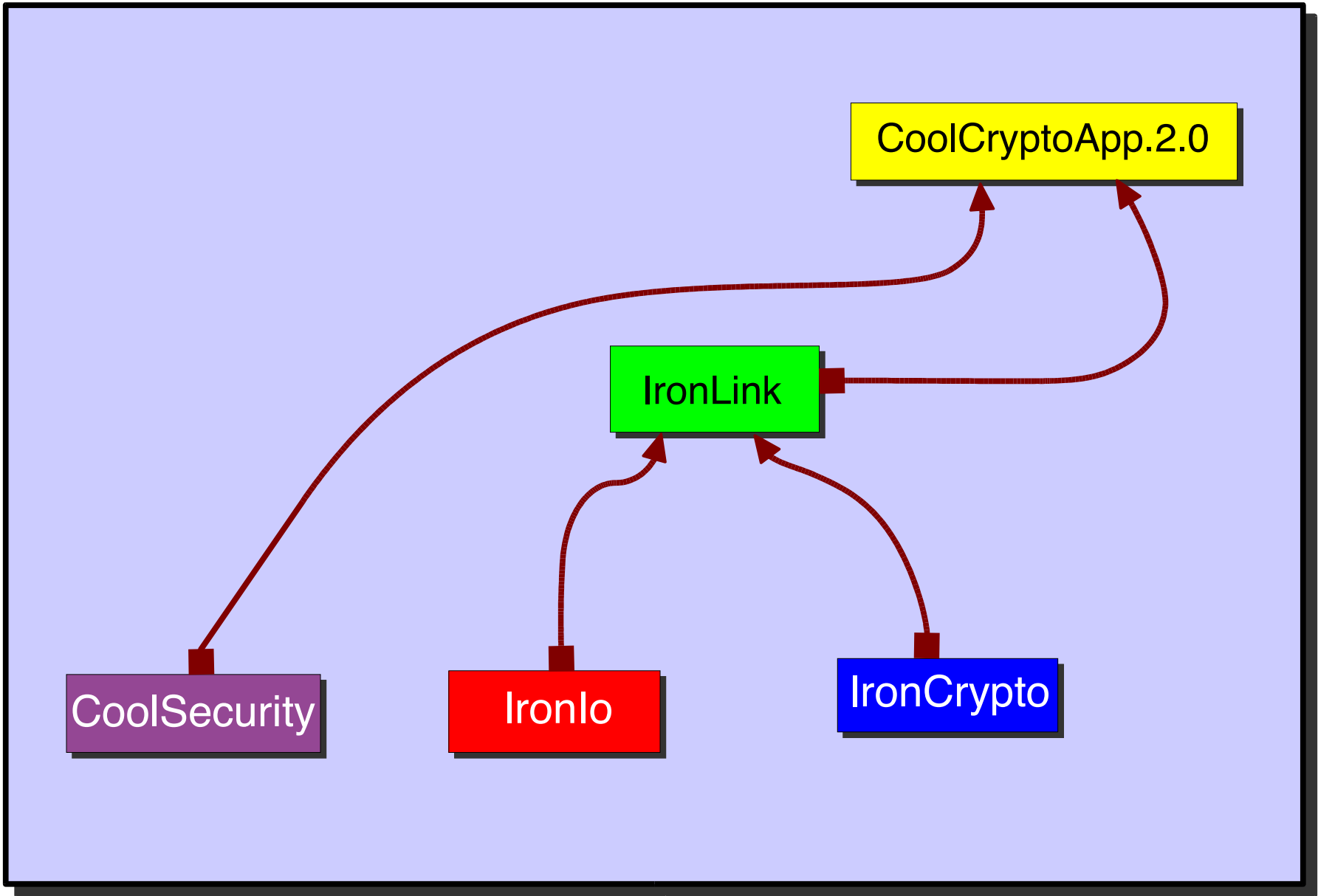


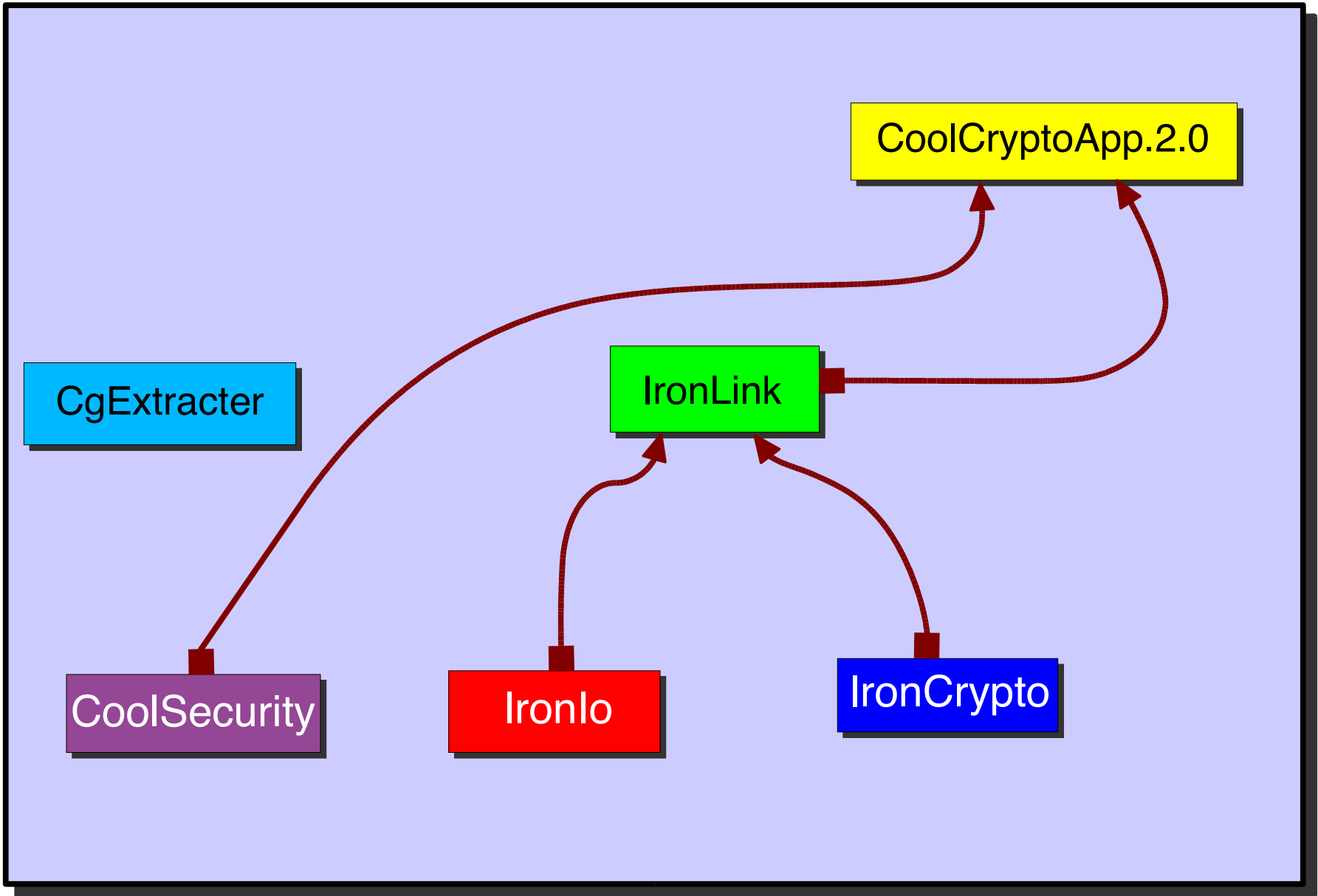


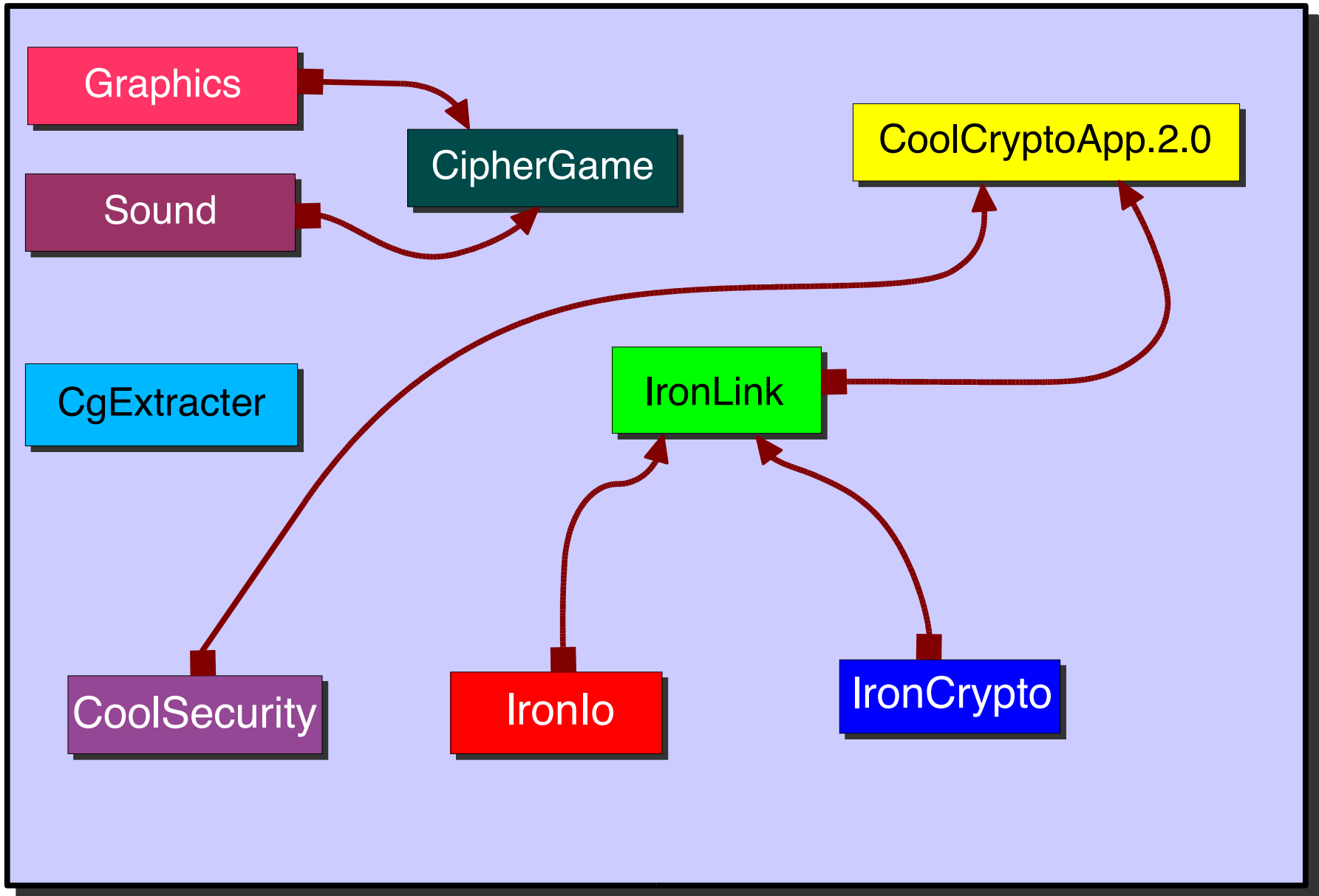


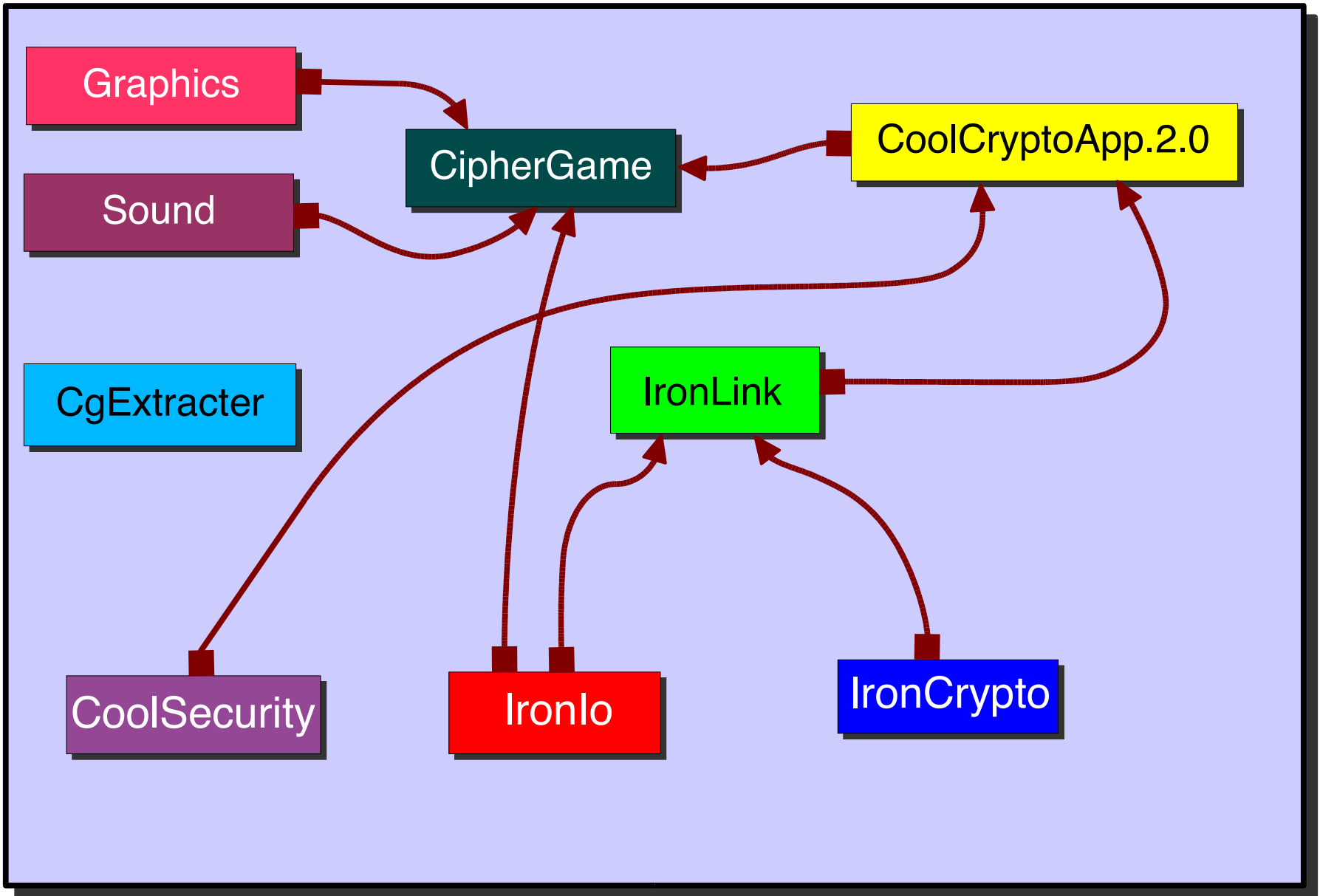
# Installation

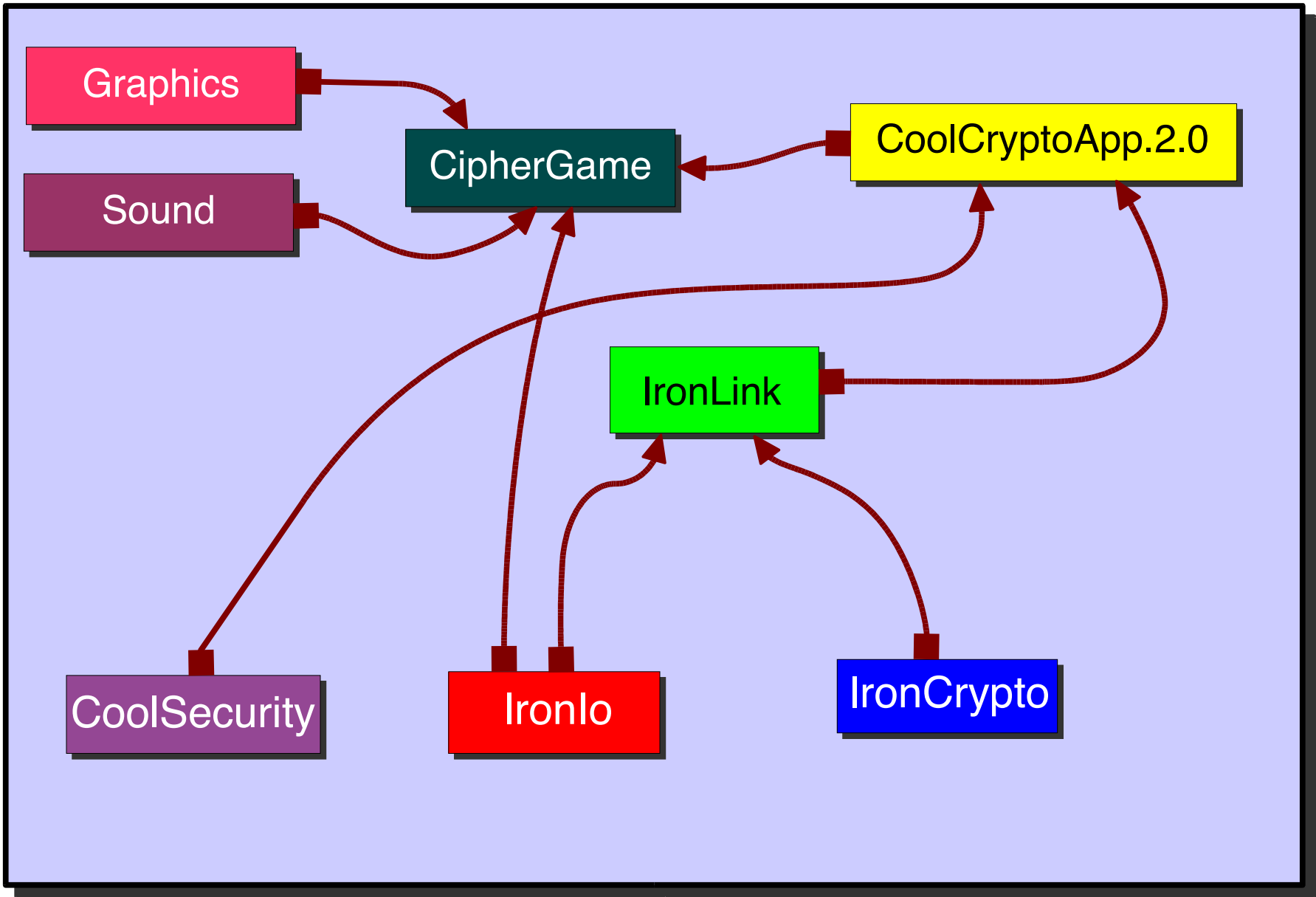
- **Extractors** are serialized and deployed
  - Self-contained components that export **installable** api
  - Deserialized in a fortress
  - **reconstitute** function is called
- Interact with a fortress via the **components** api
- Might require particular apis and components to be installed at a deployment site











# Additional advantages

- Expression of restrictions in source language
- Reduced need for package managers, rollback tools
- Bundling of additional resources (source files, etc.)
- Cross-component optimization
- Improved diagnosability
- Improved testability

# Future work

- Relationships between apis
- Integrating linking with testing
- Generalizing linking and upgrade

# Conclusion

- Deployment with robustness of static linking and resource usage and upgradability of dynamic linking
- Efficient upgrade with precise semantics
- Immutability improves resource usage
- Metacircularity improves expressiveness