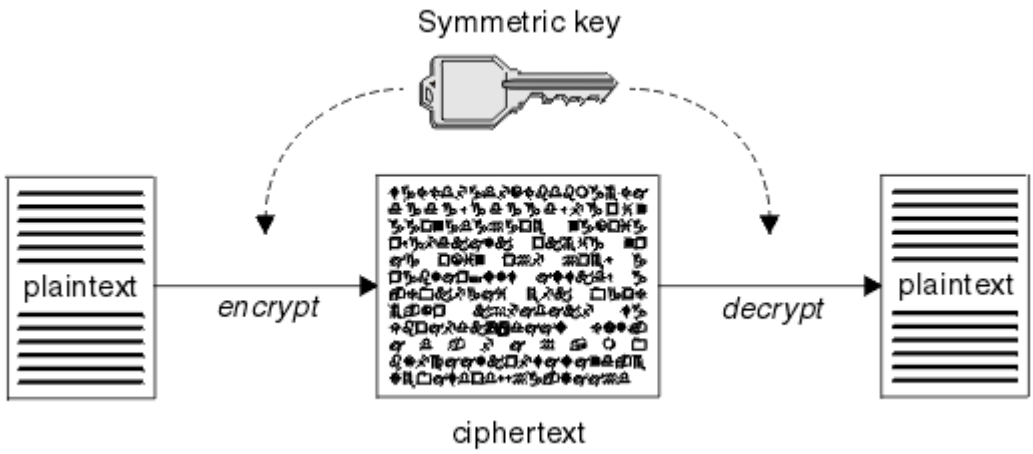
대칭키/ 공개키

대칭키 (Symmetric Key)

> 암호화와 복호화에 **같은 대칭키(암호키)**를 사용하는 알고리즘



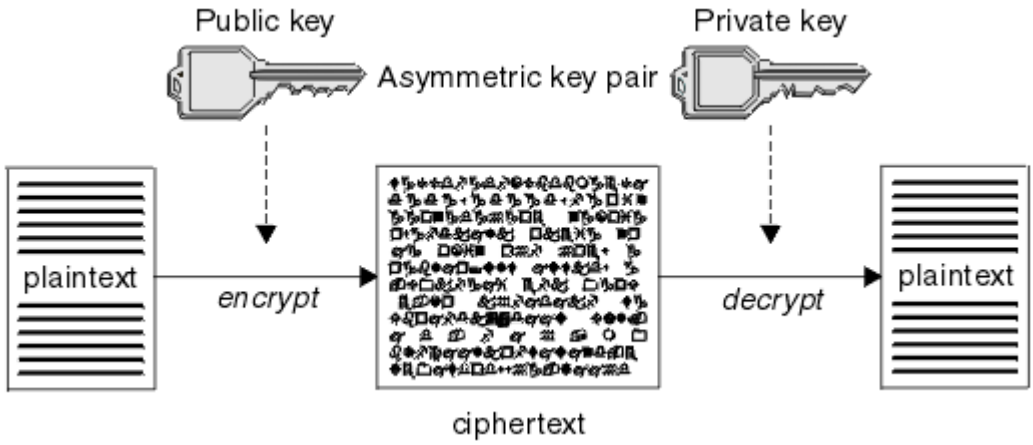
- 장점 : 공개키 암호화 방식에 비해 암호화 및 복호화 **속도가 빠르다**. 비교적 간편하다.
- 단점 : **암호화 통신을 하는 사용자끼리 같은 대칭키를 공유**해야만 한다.
  - 물리적으로 직접 만나서 전달하지 않는 한, 대칭키를 전달하는 과정에서 해킹의 위험에 노출될 수 있다.
  - 관리해야 할 키의 수가 방대해진다.
- 대표 알고리즘 : DES, 3DES, AES, SEED, ARIA 등

대칭키 암호화 시나리오

1. A는 사전에 공유된 대칭키로 데이터를 암호화하여 B에게 전송한다.
2. B는 같은 대칭키로 데이터를 복호화 한다.

공개키 (Public Key)

> 암호화와 복호화에 사용하는 **암호키를 분리한 알고리즘 (공개키 + 비밀키)**



공개키(Public Key)만 대중에게 공개하고, 암호화된 데이터는 **고유한 비밀키(Private Key)로만 복호화**할 수 있다. 이 비밀키를 가진 사용자만이 내용을 열어볼 수 있다.

자신이 가지고 있는 고유한 암호키(비밀키)로만 복호화할 수 있는 암호키(공개키)를 대중에 공개한다.

- 장점 : 수신자의 개인키로만 해독할 수 있어 **안전**하다.
- 단점 : 대칭키(Symmetric Key) 알고리즘에 비하여 **속도가 느리다.** (약 1000배)
- 대표 알고리즘 : RSA 등

공개키 암호화 시나리오

1. A가 웹상에 공개된 B의 공개키를 이용하여 평문을 암호화한다.
2. 이 암호문(CiperText)는 B가 개인적으로 가지고 있는 **B의 비밀키로만 복호화가 가능하다.** B는 자신의 비밀키로 복호화 한 평문을 확인하고, **A의 공개키로 응답을 암호화하여 A에게 보낸다.**
3. A는 A의 비밀키로 암호화된 응답 문을 복호화한다.

대칭키와 공개키 혼합 (하이브리드 방식)

> SSL(Secure Socket Layer) 탄생의 시초가 되었음

대칭키를 주고받을 때만 공개키 암호화 방식을 사용하고, 이후에는 **계속 대칭키 암호화 방식**으로 통신한다!

1. A가 B의 공개키로 암호화 통신에 사용할 대칭키를 암호화하여 B에게 보낸다.
2. B는 암호문을 받아, 자신의 비밀키로 복호화한다.
3. B는 A로부터 얻은 대칭키로 A에게 보낼 평문을 암호화하여 A에게 보낸다.
4. A는 자신의 대칭키로 암호문을 복호화한다.
5. 계속 대칭키로 암호화 통신을 한다.

SSL은 웹서버와 브라우저 간의 통신을 암호화 해서 중간에 누가 가로채더라도 내용을 알 수 없게 한다. SSL이 작용되면 https:// 를 사용하여 웹서버에 접근하게 된다.

대칭키 암호화는 서버와 클라이언트가 같은 키를 사용해야 하므로 키를 공유하는데 문제가 있고, 비대칭키 암호화는 공개키를 배포함으로써 키 공유 문제는 해결되지만, 처리속도가 느린 문제가 있다. **따라서 HTTPS 통신에서 실제 전송되는 데이터의 암호화에는 대칭키 암호화 방식을 사용하고, 키 교환에는 비대칭키 암호화를 사용하여 이러한 문제를 해결하고 있다.**