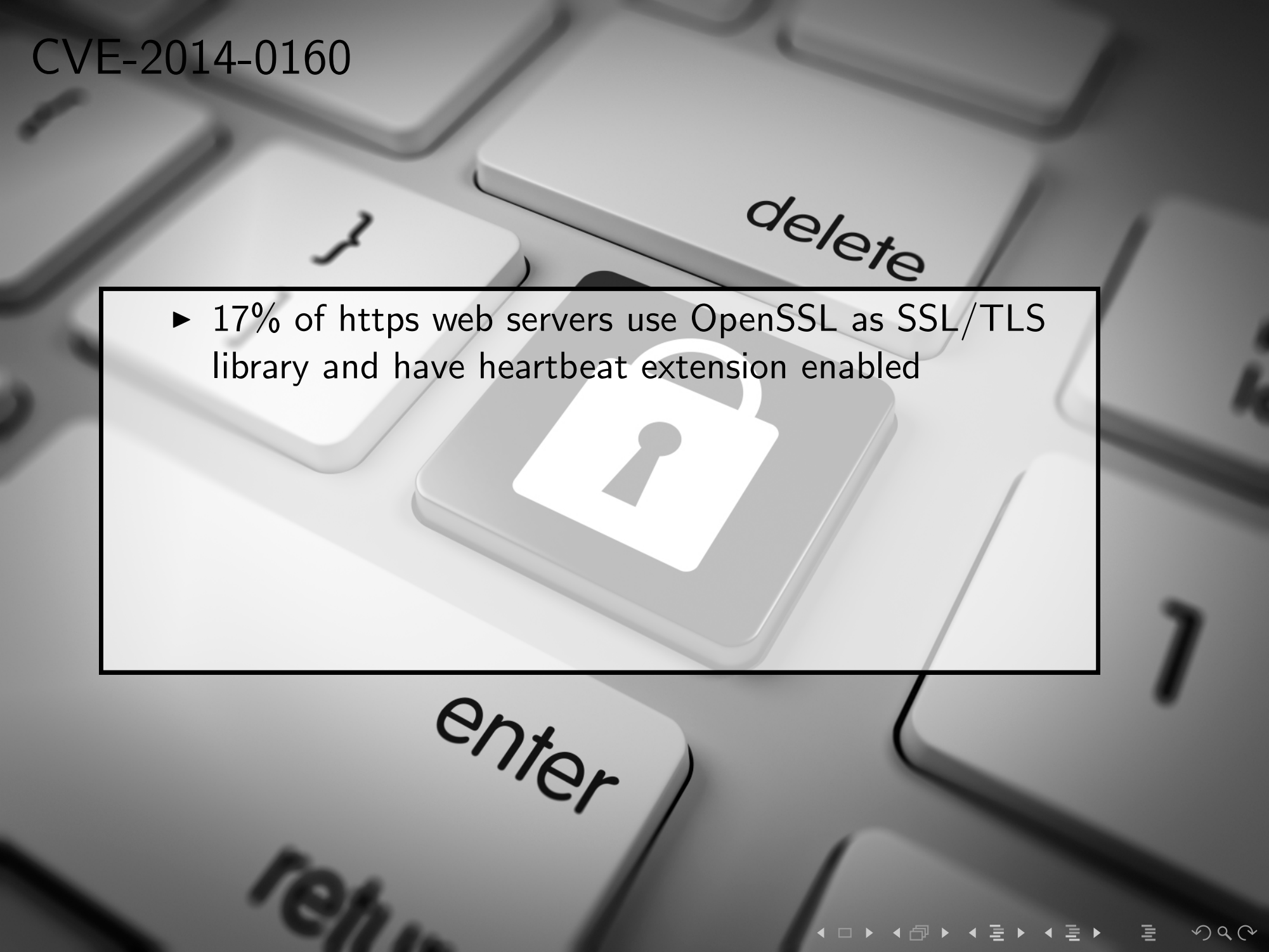
# LibreSSL

Giovanni Bechis
<giovanni@openbsd.org>

International Crypto Module Conference 2016

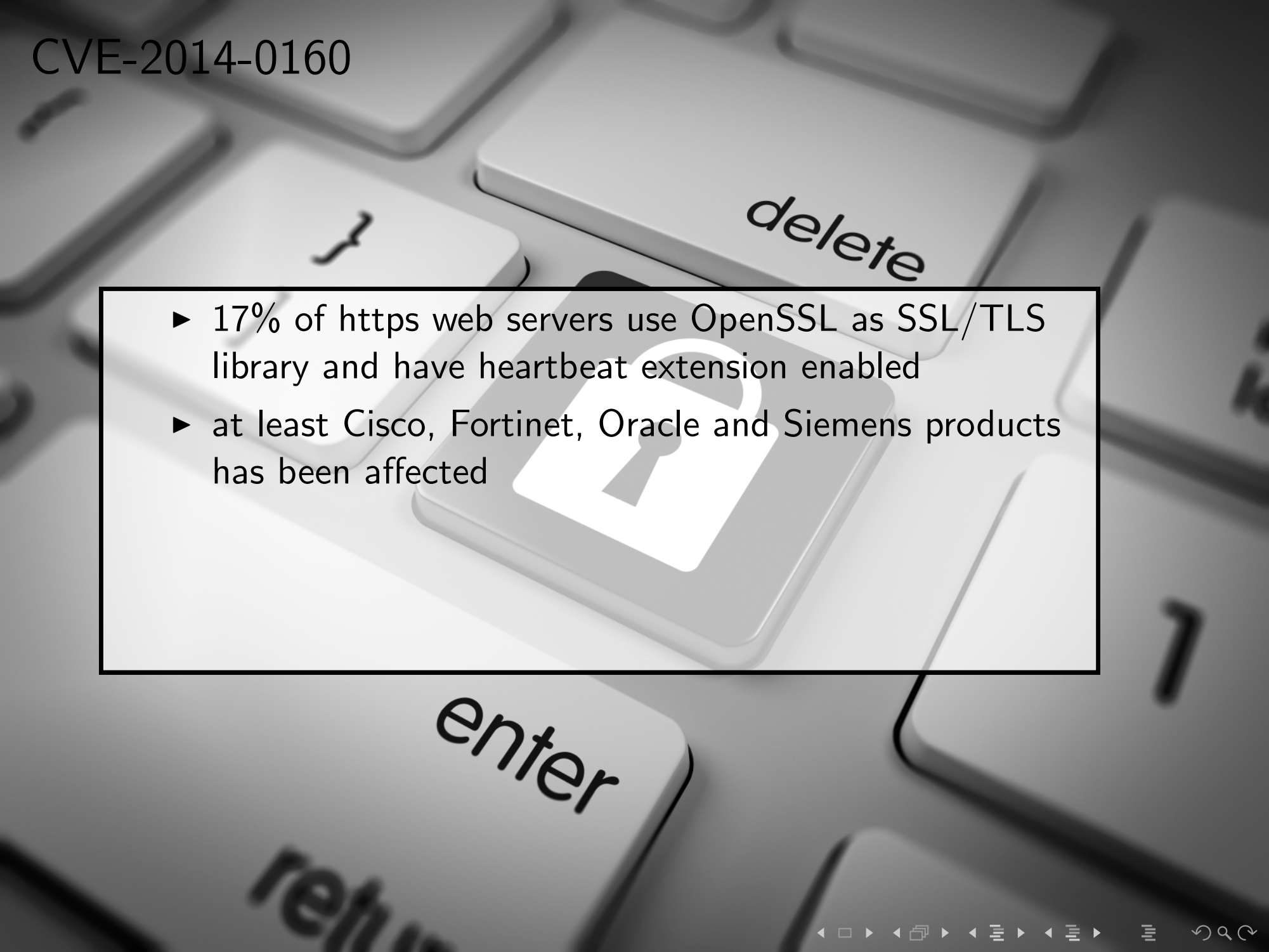
SNB Information Technology & Web Solutions

# CVE-2014-0160



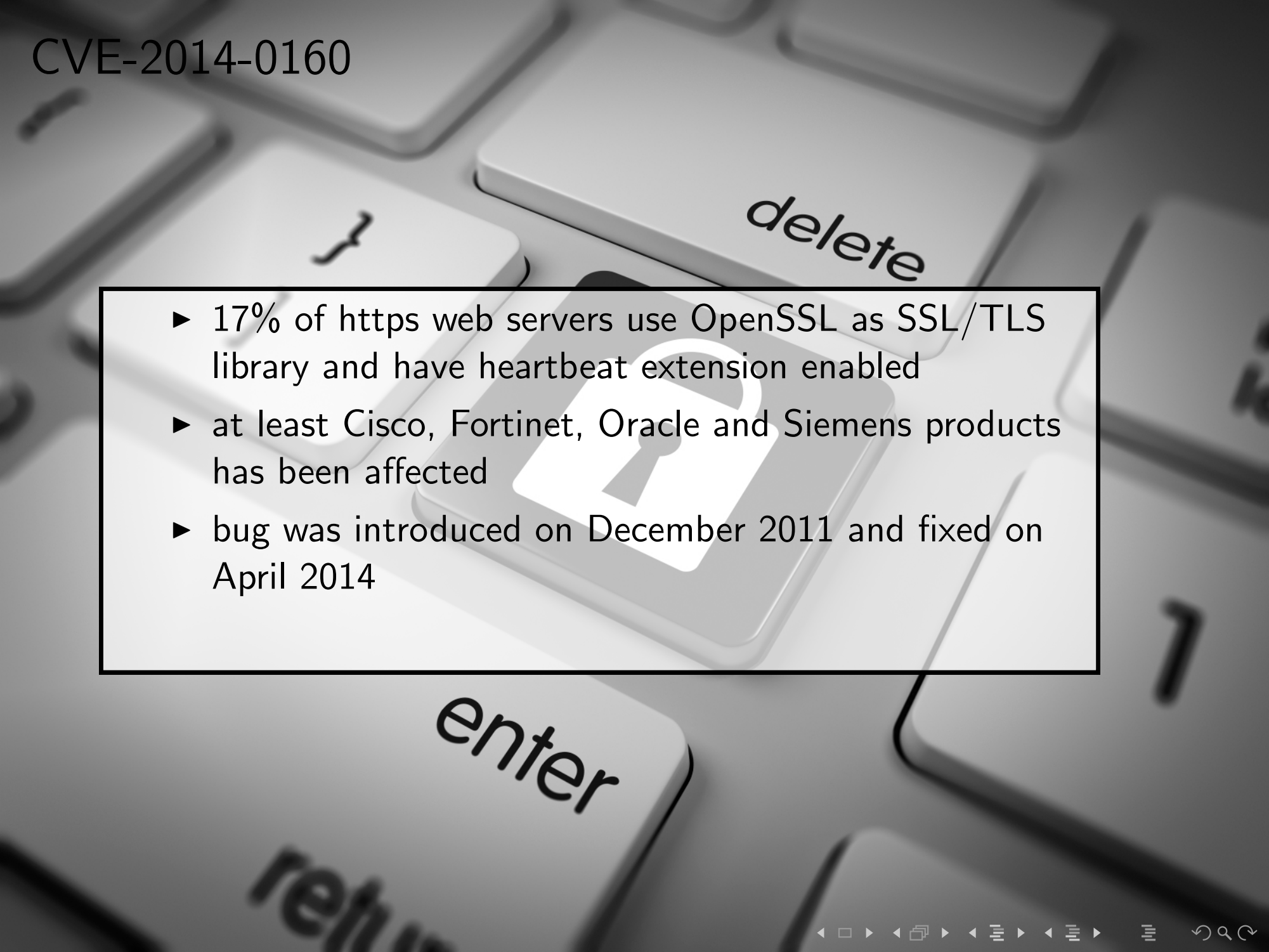

- 17% of https web servers use OpenSSL as SSL/TLS library and have heartbeat extension enabled

# CVE-2014-0160



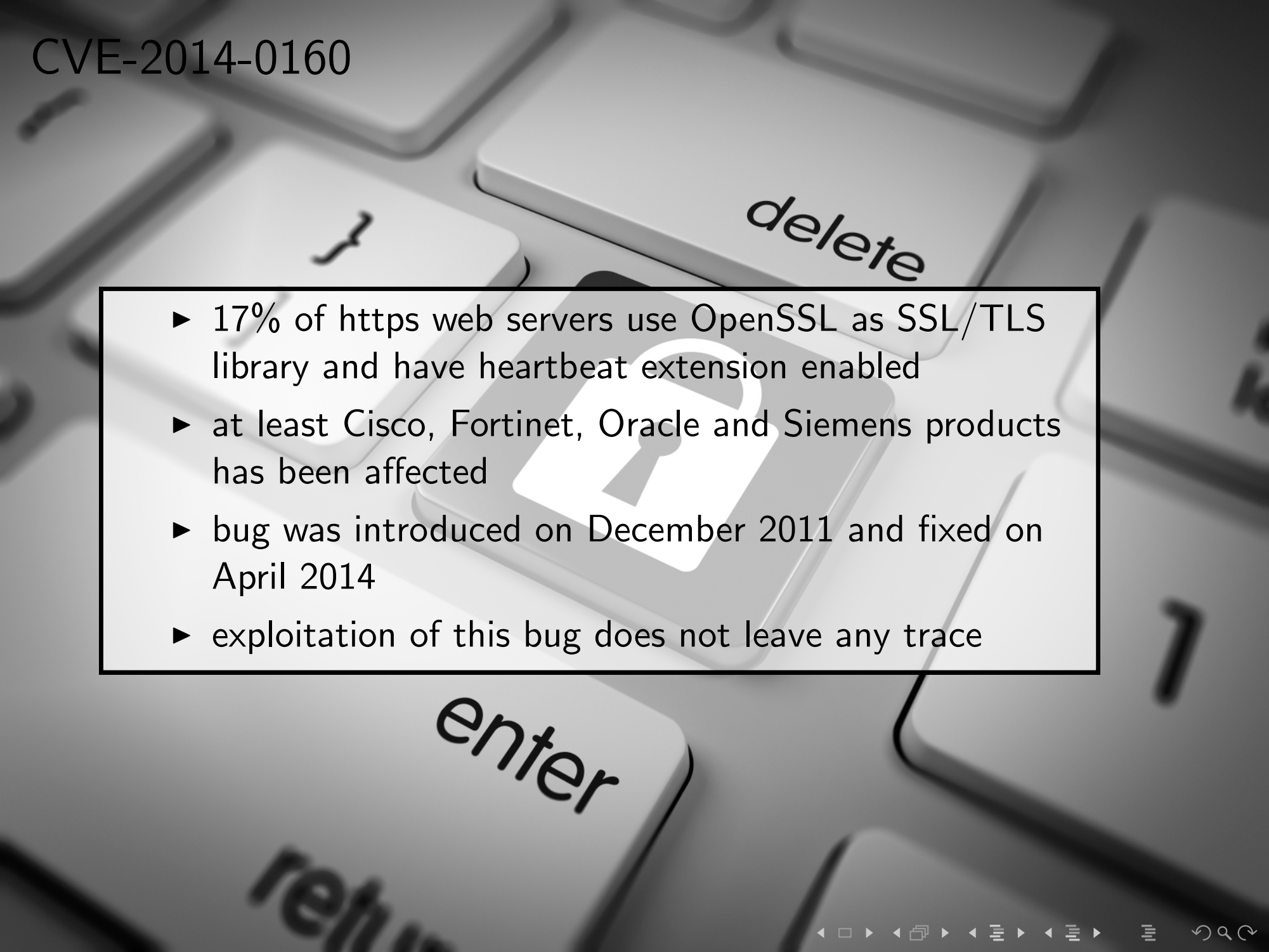

- 17% of https web servers use OpenSSL as SSL/TLS library and have heartbeat extension enabled
- at least Cisco, Fortinet, Oracle and Siemens products has been affected

# CVE-2014-0160



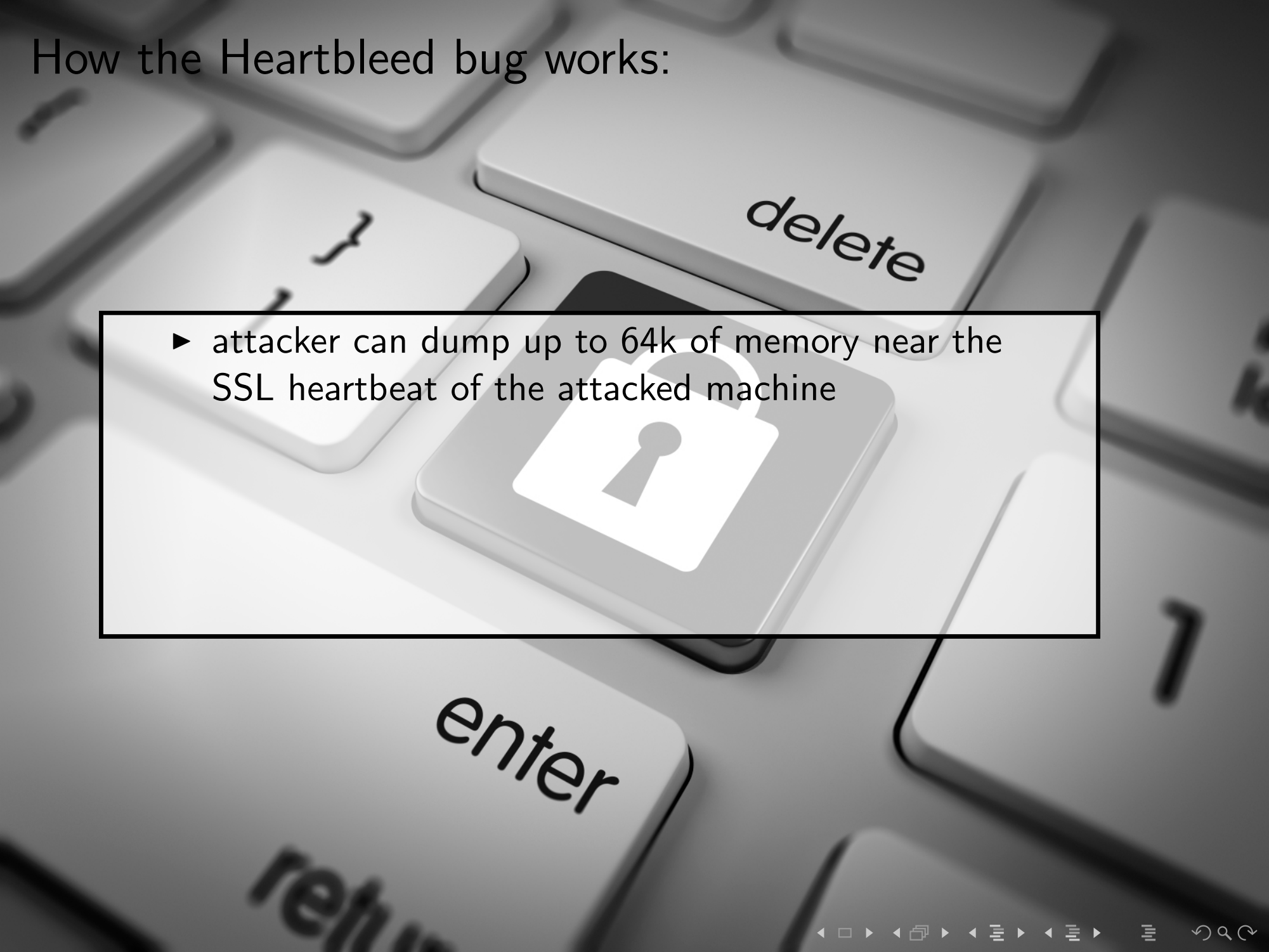

- 17% of https web servers use OpenSSL as SSL/TLS library and have heartbeat extension enabled
- at least Cisco, Fortinet, Oracle and Siemens products has been affected
- bug was introduced on December 2011 and fixed on April 2014

# CVE-2014-0160



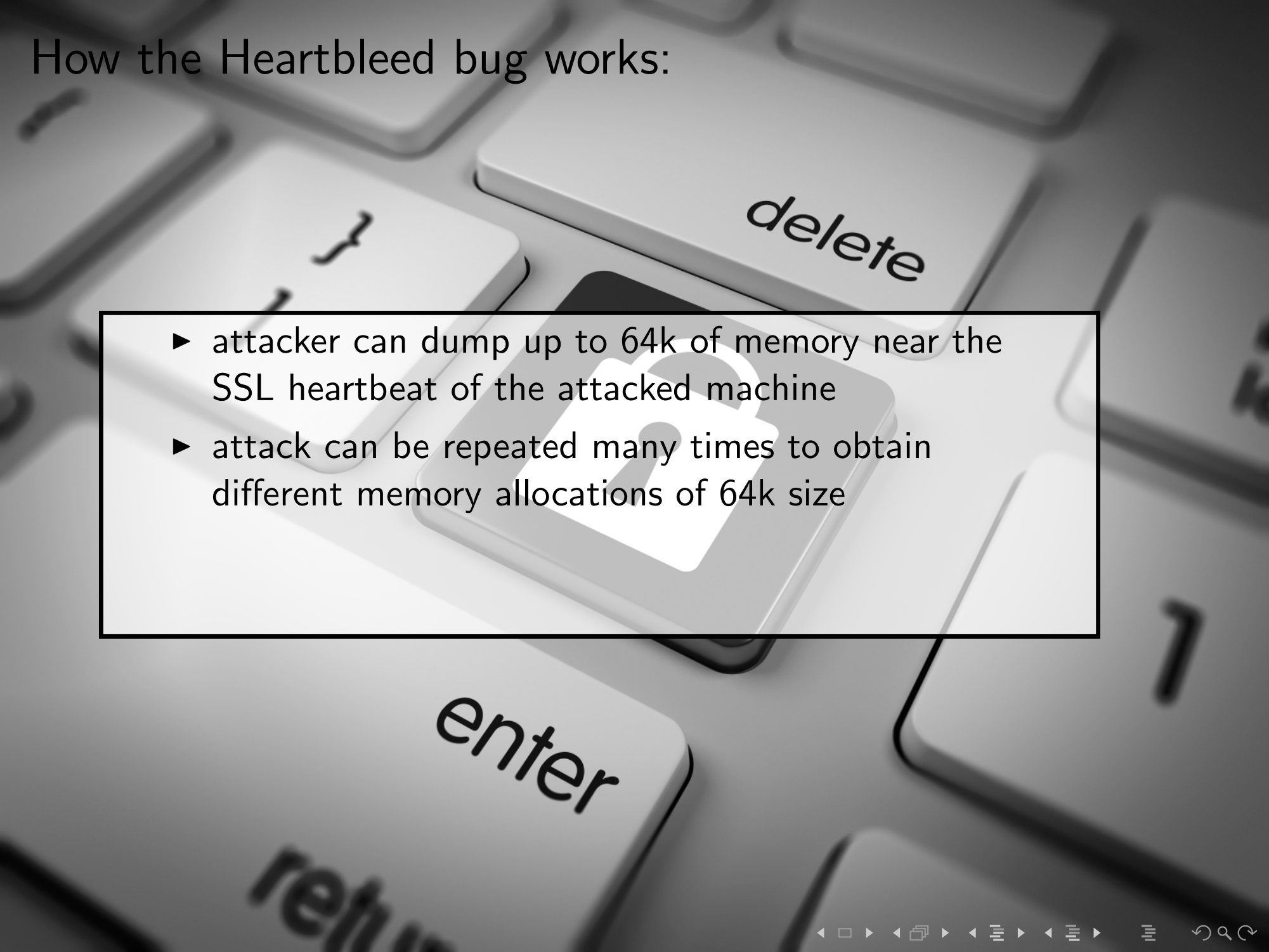

- 17% of https web servers use OpenSSL as SSL/TLS library and have heartbeat extension enabled
- at least Cisco, Fortinet, Oracle and Siemens products has been affected
- bug was introduced on December 2011 and fixed on April 2014
- exploitation of this bug does not leave any trace

# How the Heartbleed bug works:




- attacker can dump up to 64k of memory near the SSL heartbeat of the attacked machine

# How the Heartbleed bug works:




- attacker can dump up to 64k of memory near the SSL heartbeat of the attacked machine
- attack can be repeated many times to obtain different memory allocations of 64k size

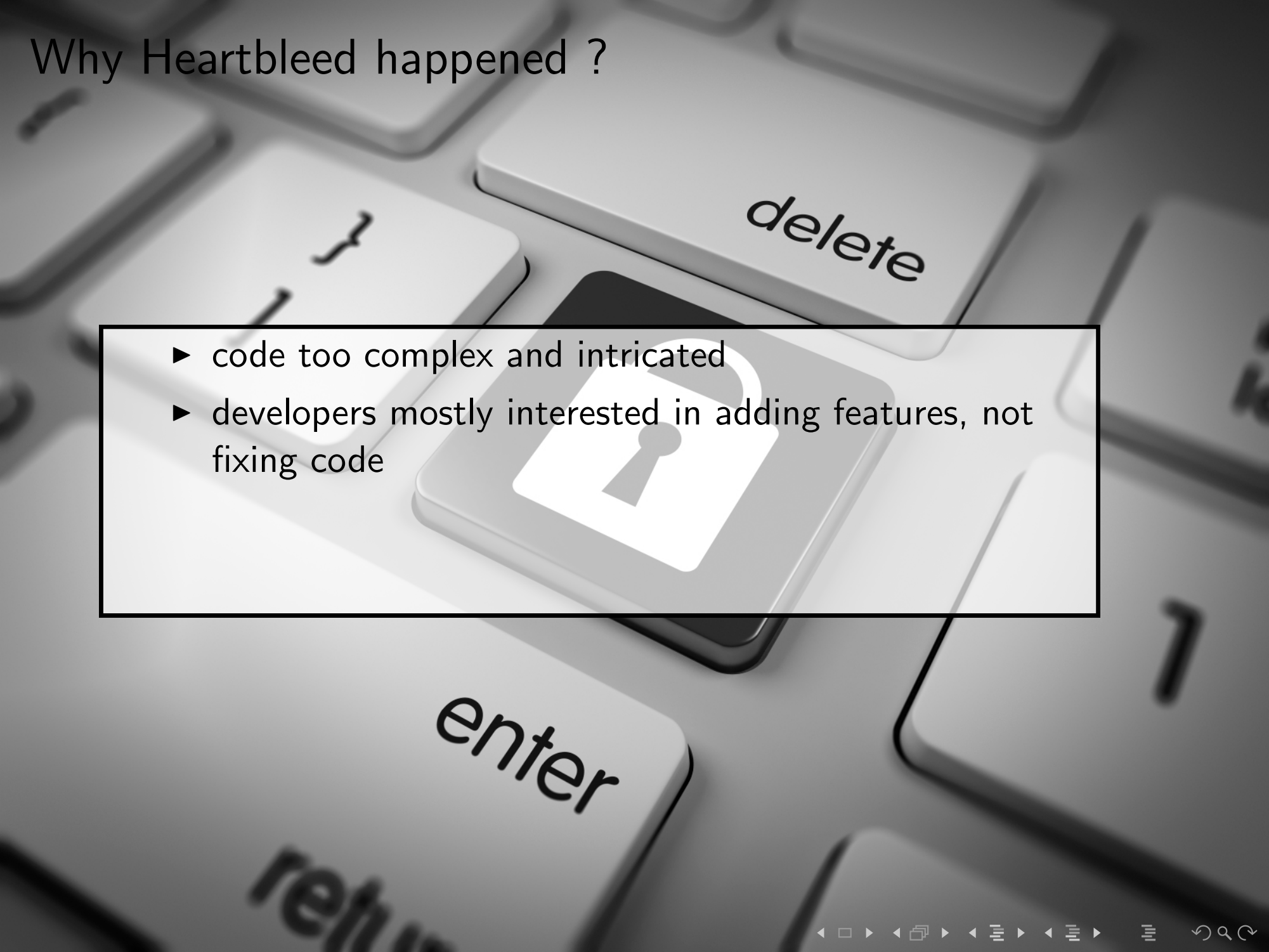
# How the Heartbleed bug works:

- attacker can dump up to 64k of memory near the SSL heartbeat of the attacked machine
- attack can be repeated many times to obtain different memory allocations of 64k size
- memory stolen could reveal any kind of data: passwords, credit card numbers, personal data, ...

# Why Heartbleed happened ?

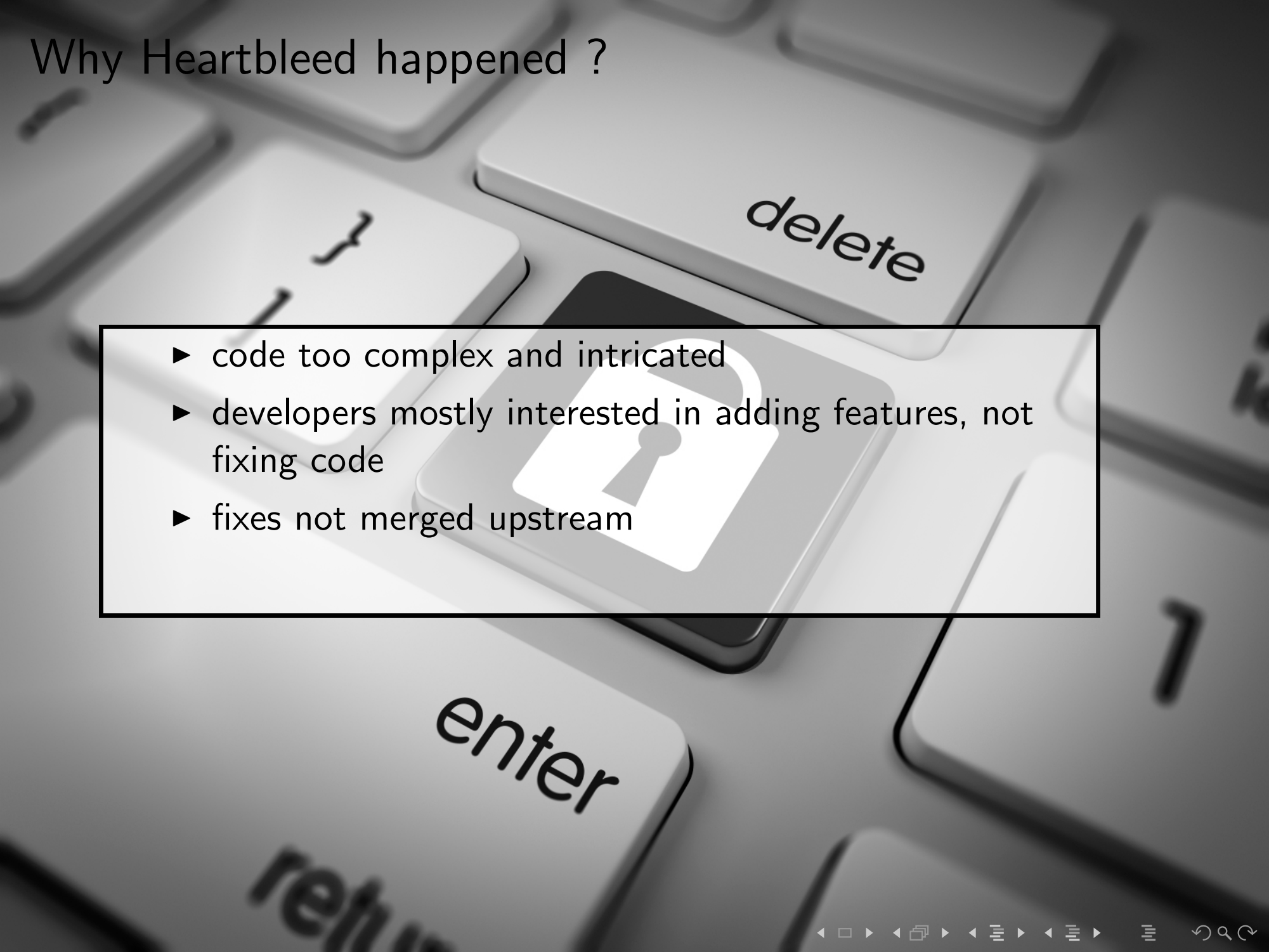

- code too complex and intricated

# Why Heartbleed happened ?



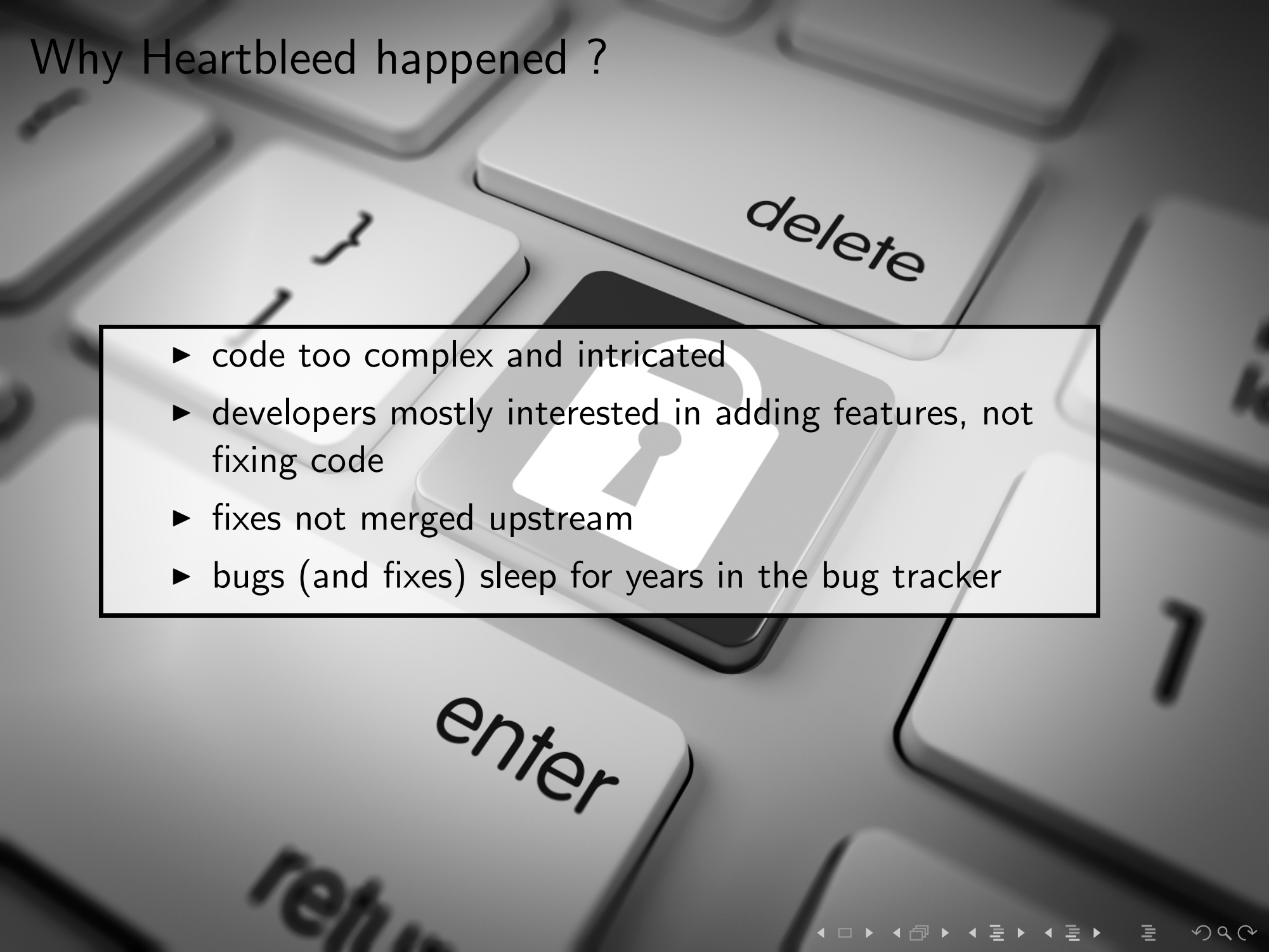

- code too complex and intricated
- developers mostly interested in adding features, not fixing code

# Why Heartbleed happened ?




- code too complex and intricated
- developers mostly interested in adding features, not fixing code
- fixes not merged upstream

# Why Heartbleed happened ?




- code too complex and intricated
- developers mostly interested in adding features, not fixing code
- fixes not merged upstream
- bugs (and fixes) sleep for years in the bug tracker

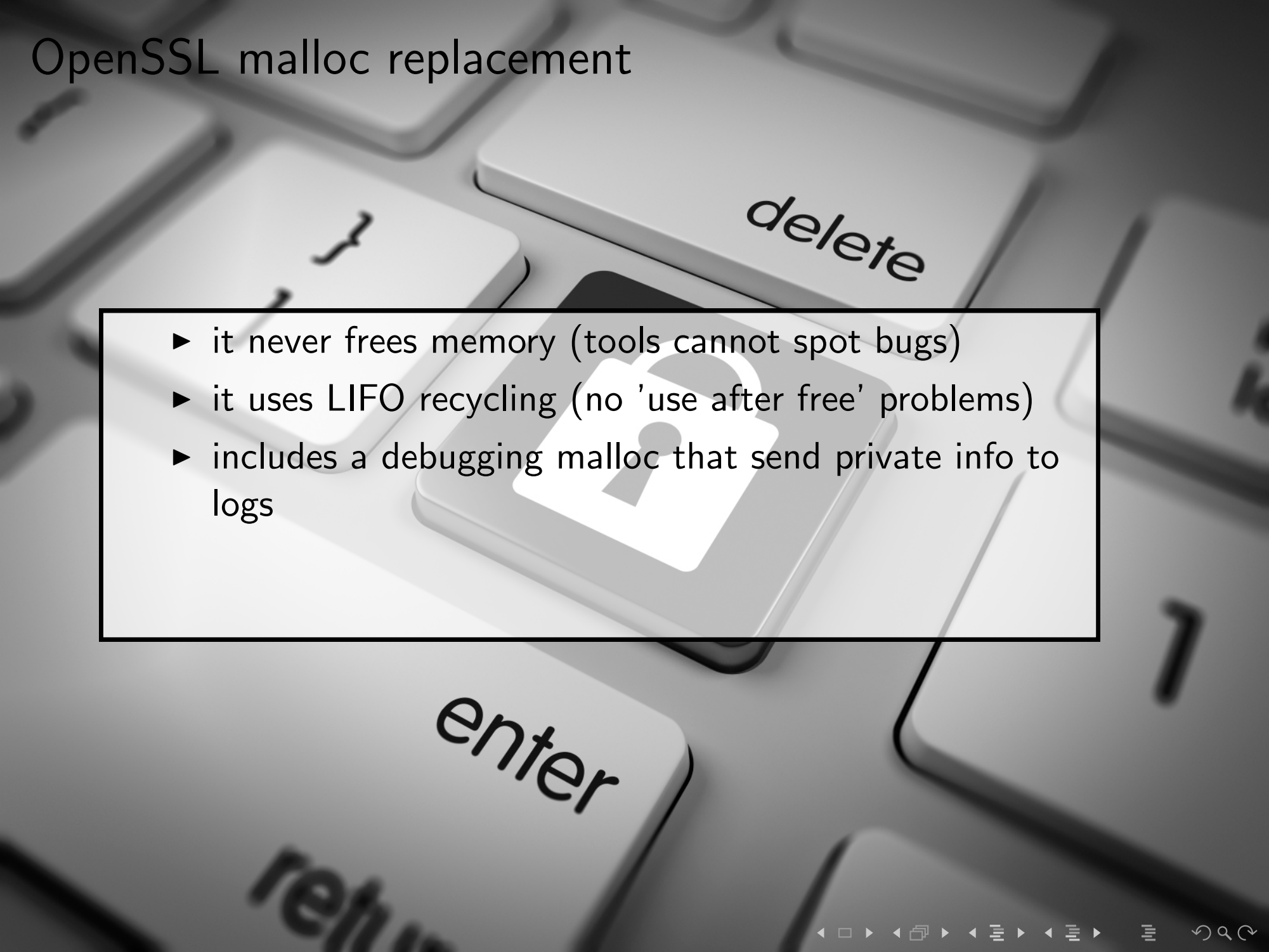
# OpenSSL malloc replacement


- it never frees memory (tools cannot spot bugs)

# OpenSSL malloc replacement


- it never frees memory (tools cannot spot bugs)
- it uses LIFO recycling (no 'use after free' problems)

# OpenSSL malloc replacement




- it never frees memory (tools cannot spot bugs)
- it uses LIFO recycling (no 'use after free' problems)
- includes a debugging malloc that send private info to logs

# OpenSSL malloc replacement


- it never frees memory (tools cannot spot bugs)
- it uses LIFO recycling (no 'use after free' problems)
- includes a debugging malloc that send private info to logs
- includes the ability to replace the malloc/free at runtime

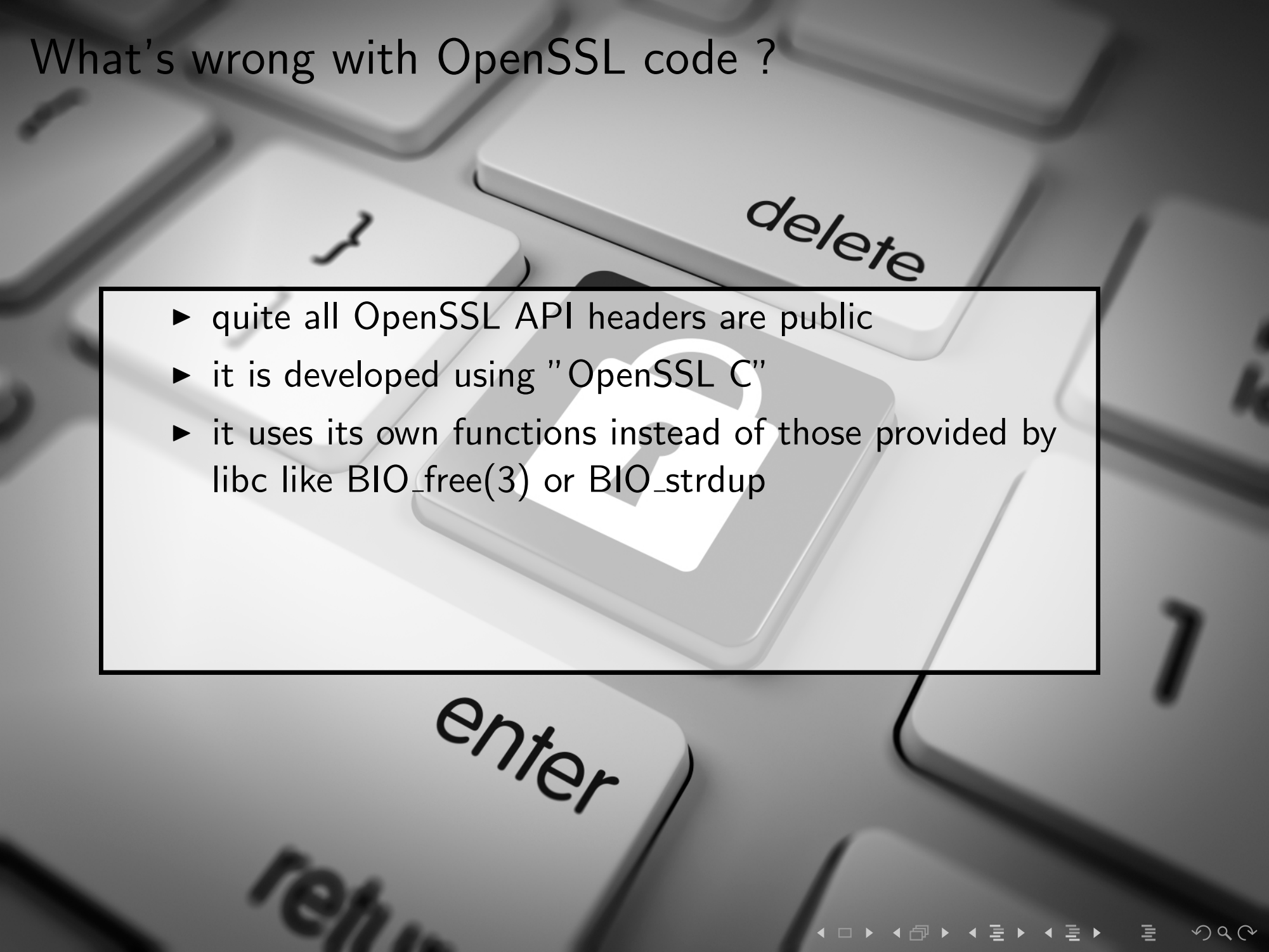
# What's wrong with OpenSSL code ?

- quite all OpenSSL API headers are public

# What's wrong with OpenSSL code ?


- quite all OpenSSL API headers are public
- it is developed using "OpenSSL C"

# What's wrong with OpenSSL code ?




- quite all OpenSSL API headers are public
- it is developed using "OpenSSL C"
- it uses its own functions instead of those provided by libc like BIO_free(3) or BIO_strdup

# What's wrong with OpenSSL code ?

- quite all OpenSSL API headers are public
- it is developed using "OpenSSL C"
- it uses its own functions instead of those provided by libc like BIO_free(3) or BIO_strdup
- it has strange compile options (in OpenSSL both NO_OLD_ASN1 and NO_ASN1_OLD compile options are present but their meaning is slightly different)

# what's wrong with OpenSSL ?

```
#include "des_locl.h"

/* HAS BUGS! DON'T USE - this is only present for use in des.c */
void DES_3cbc_encrypt(DES_cblock *input, DES_cblock *output, long length,
             DES_key_schedule ks1, DES_key_schedule ks2, DES_cblock *iv1,
             DES_cblock *iv2, int enc)
```

# what's wrong with OpenSSL ?

```
===================================================================
RCS file: /var/cvs/src/lib/libssl/src/apps/Attic/s_socket.c,v
retrieving revision 1.31
retrieving revision 1.32
diff -u -p -r1.31 -r1.32
--- apps/s_socket.c     19 Apr 2014 13:13:01 -0000      1.31
+++ apps/s_socket.c     19 Apr 2014 16:38:04 -0000      1.32
@@ -77,7 +77,6 @@
 #ifndef OPENSSL_NO_SOCK


-static struct hostent *GetHostByName(char *name);
 static int ssl_sock_init(void);
 static int init_server(int *sock, int port, int type);
 static int init_server_long(int *sock, int port, char *ip, int type);
@@ -296,7 +295,7 @@ redoit:
                        return (0);
                }

-               h2 = GetHostByName(*host);
+               h2 = gethostbyname(*host);
                if (h2 == NULL) {
                        BIO_printf(bio_err, "gethostbyname failure\n");
```

# LibreSSL


- young project, development started on April 2014

# LibreSSL


- young project, development started on April 2014
- mostly developed by OpenBSD team

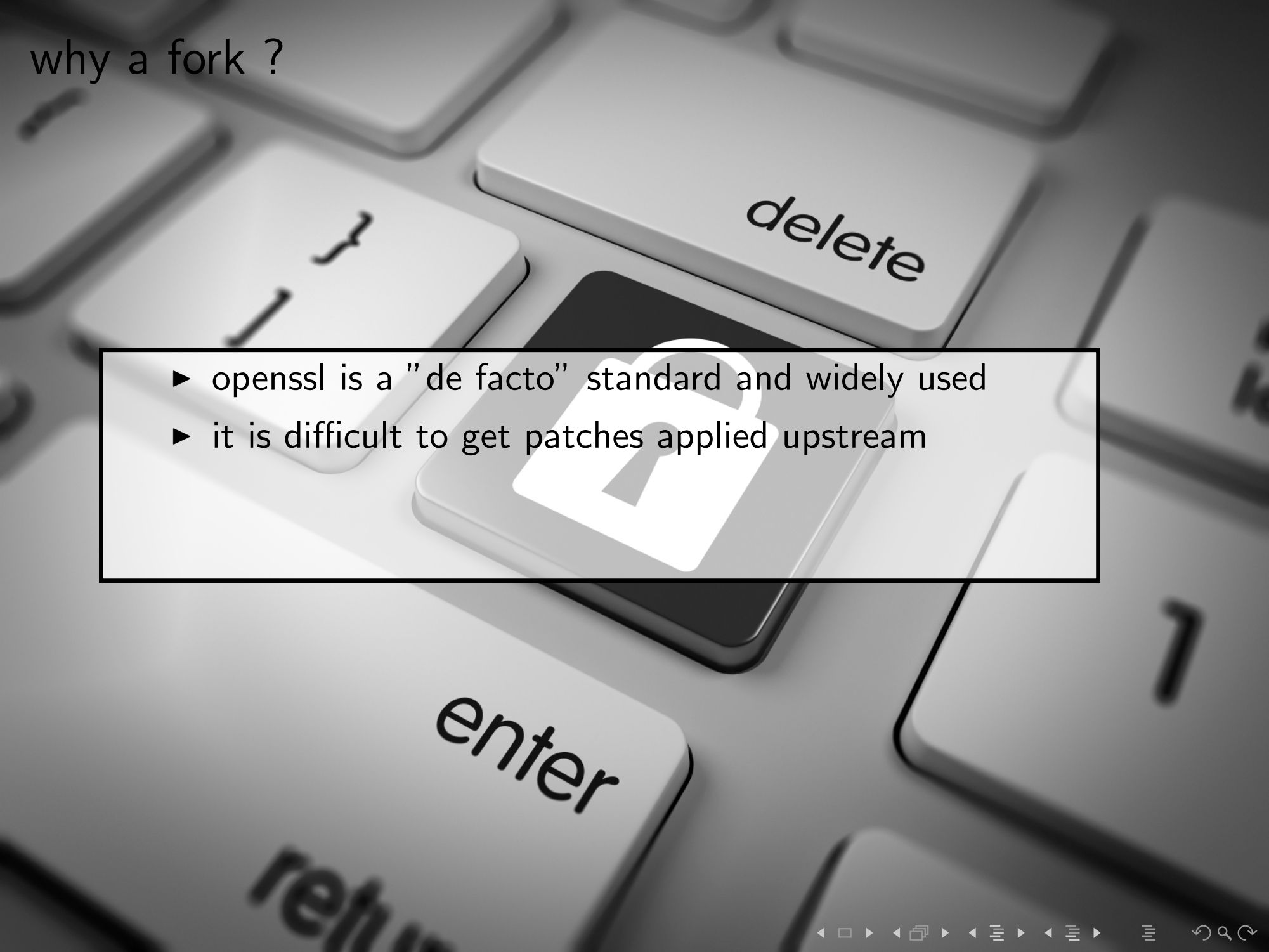
# LibreSSL


- young project, development started on April 2014
- mostly developed by OpenBSD team
- forked from OpenSSL 1.0.1g

# why a fork ?

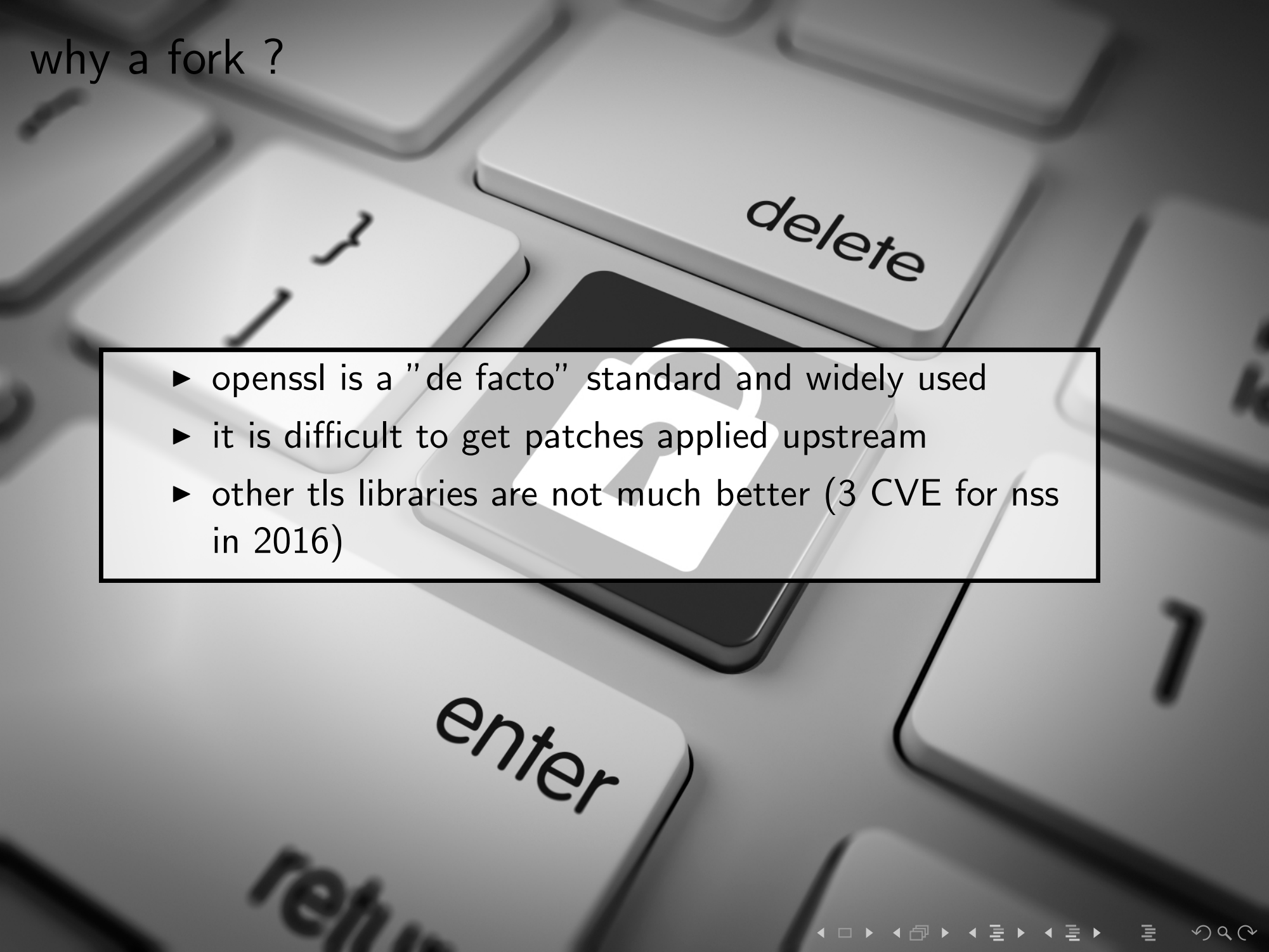

- openssl is a "de facto" standard and widely used

# why a fork ?




- openssl is a "de facto" standard and widely used
- it is difficult to get patches applied upstream

# why a fork ?

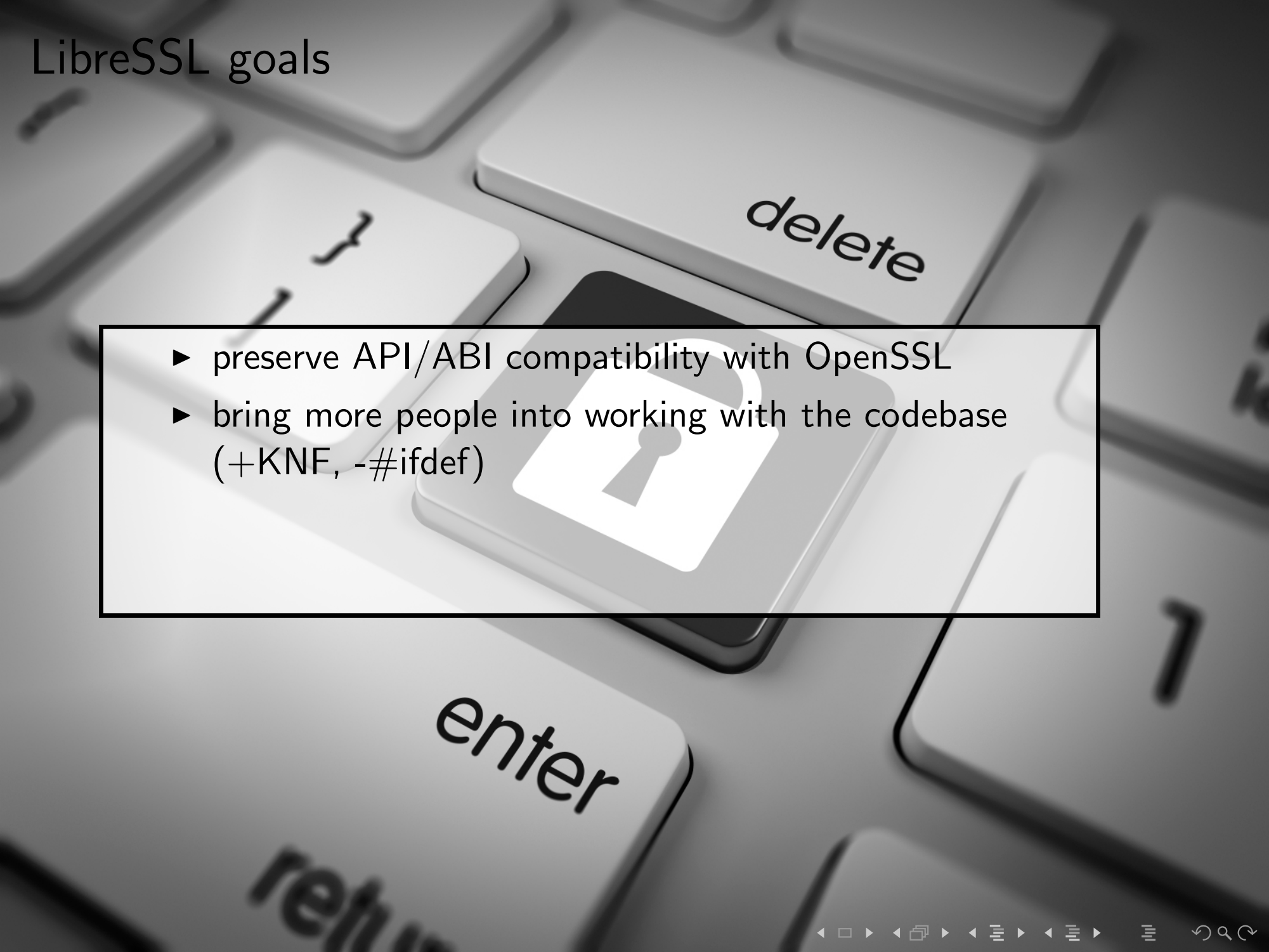



- openssl is a "de facto" standard and widely used
- it is difficult to get patches applied upstream
- other tls libraries are not much better (3 CVE for nss in 2016)

# LibreSSL goals

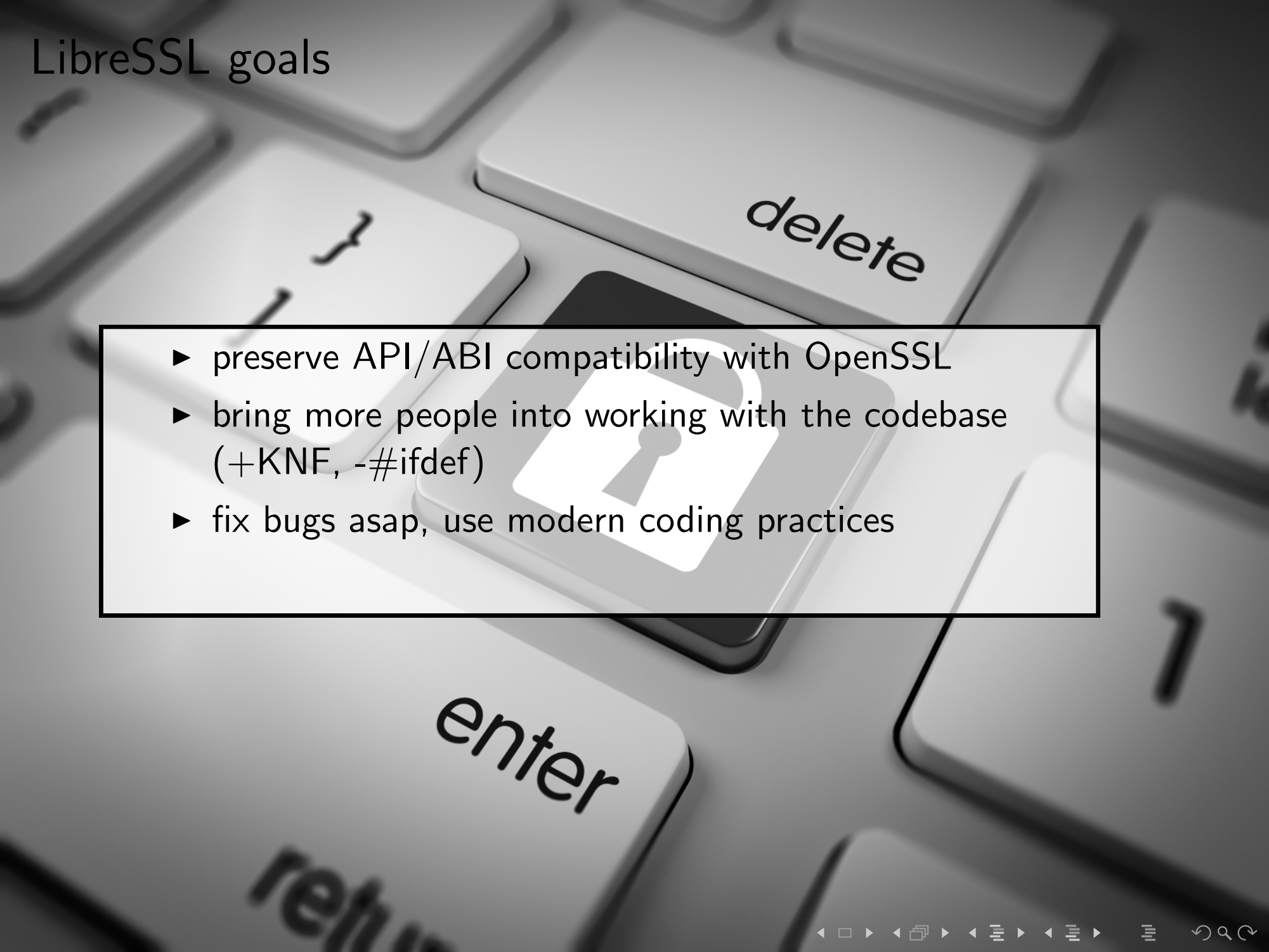

- preserve API/ABI compatibility with OpenSSL

# LibreSSL goals



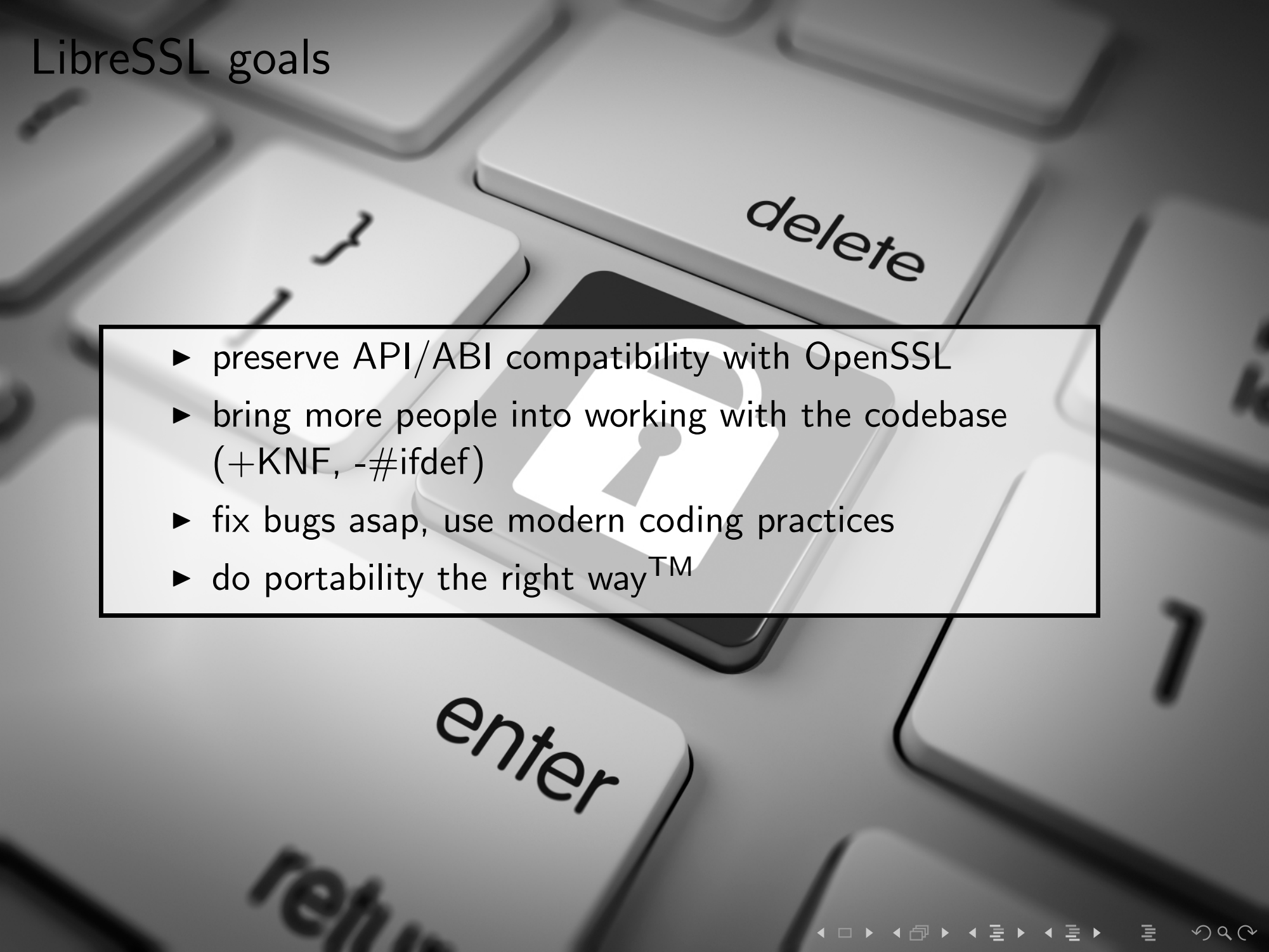

- preserve API/ABI compatibility with OpenSSL
- bring more people into working with the codebase (+KNF, -#ifdef)

# LibreSSL goals




- preserve API/ABI compatibility with OpenSSL
- bring more people into working with the codebase (+KNF, -#ifdef)
- fix bugs asap, use modern coding practices

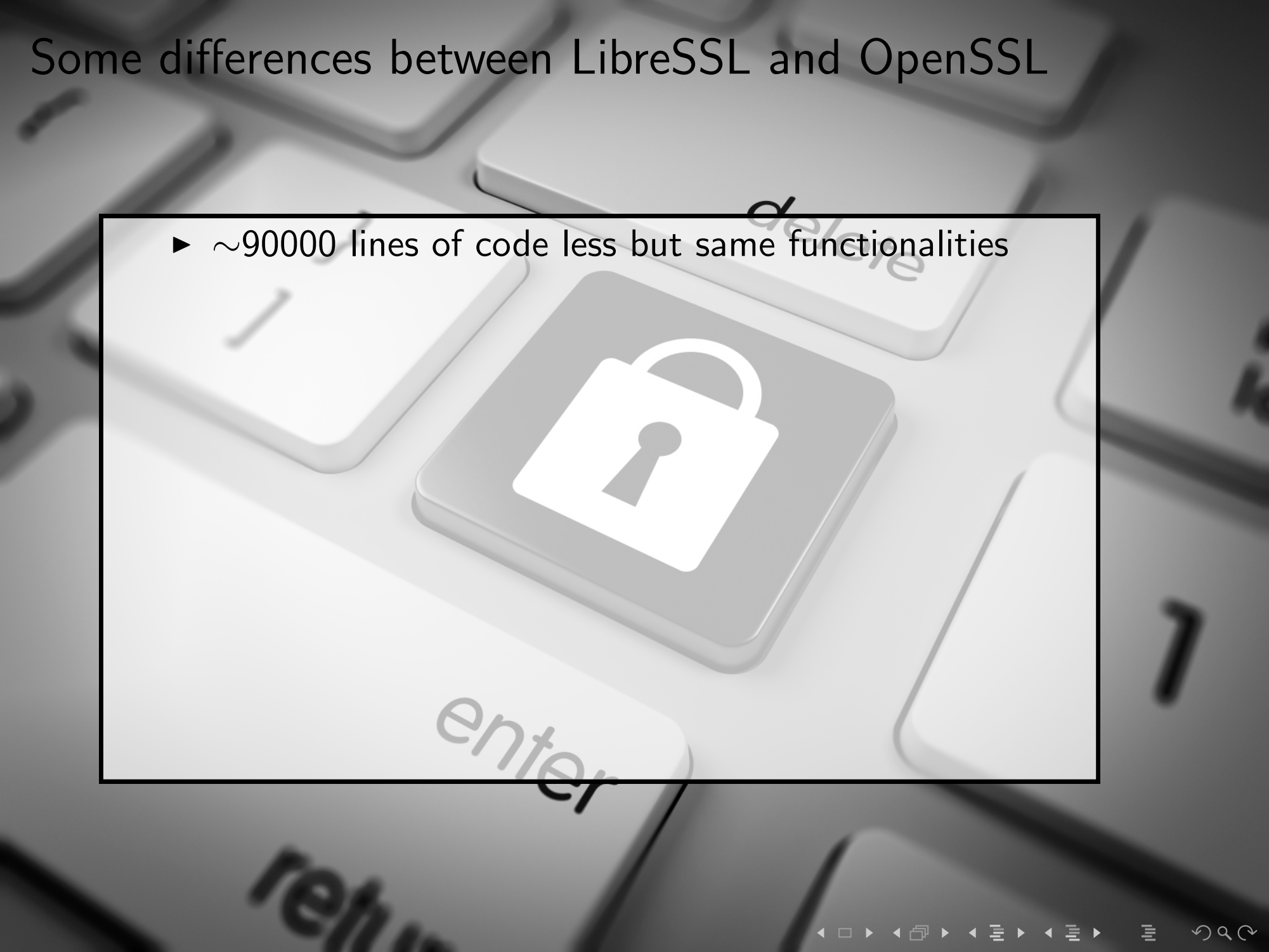
# LibreSSL goals




- preserve API/ABI compatibility with OpenSSL
- bring more people into working with the codebase (+KNF, -#ifdef)
- fix bugs asap, use modern coding practices
- do portability the right way™

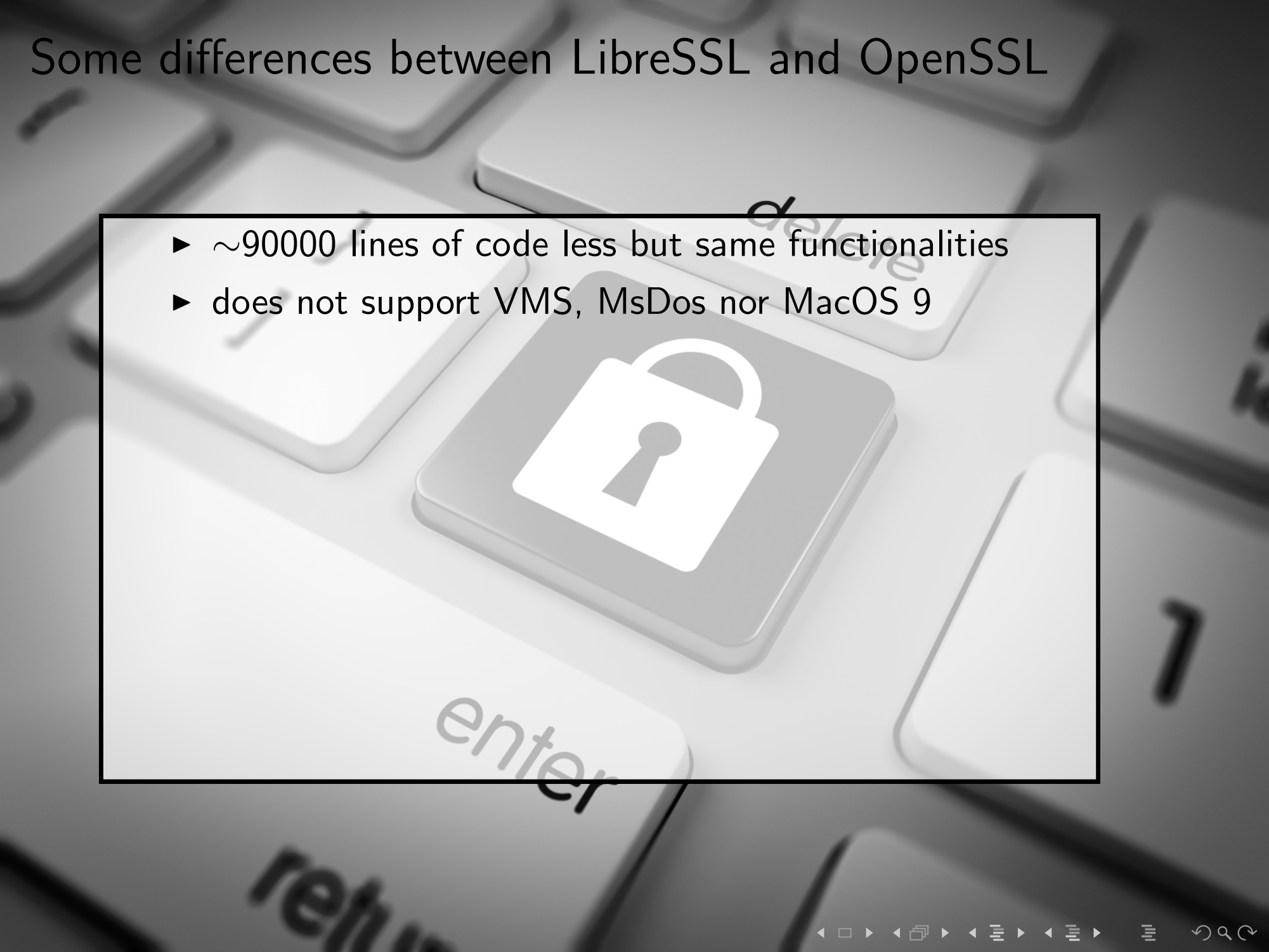
# Some differences between LibreSSL and OpenSSL

- ∼90000 lines of code less but same functionalities



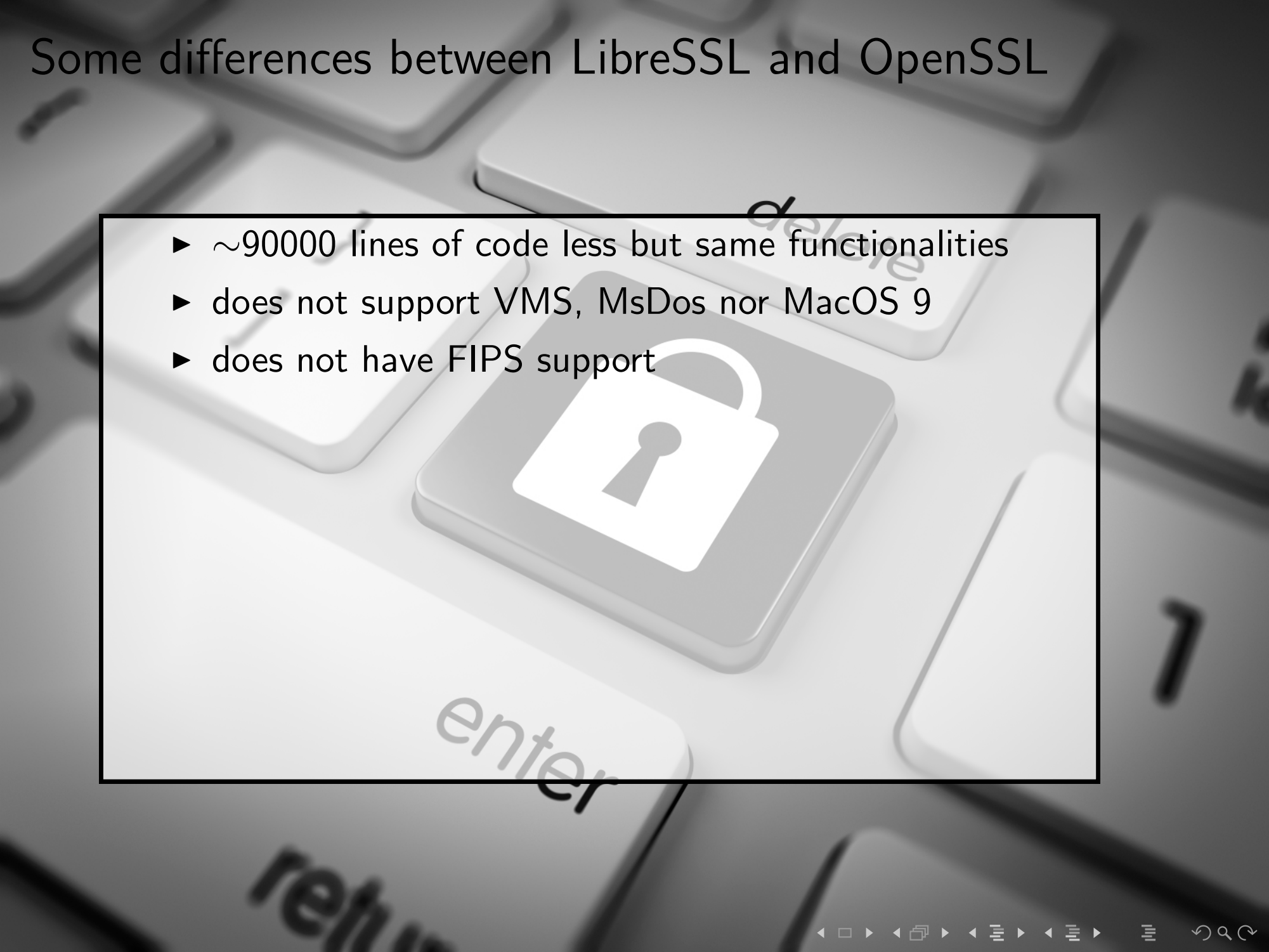
# Some differences between LibreSSL and OpenSSL

- ∼90000 lines of code less but same functionalities
- does not support VMS, MsDos nor MacOS 9



# Some differences between LibreSSL and OpenSSL

- ∼90000 lines of code less but same functionalities
- does not support VMS, MsDos nor MacOS 9
- does not have FIPS support



# Some differences between LibreSSL and OpenSSL

- ∼90000 lines of code less but same functionalities
- does not support VMS, MsDos nor MacOS 9
- does not have FIPS support
- different set of ciphers (-SRP, +ChaCha, +poly1305)

# Some differences between LibreSSL and OpenSSL

- ~90000 lines of code less but same functionalities
- does not support VMS, MsDos nor MacOS 9
- does not have FIPS support
- different set of ciphers (-SRP, +ChaCha, +poly1305)
- BIO_* functions do the right thing™

# Some differences between LibreSSL and OpenSSL

- ~90000 lines of code less but same functionalities
- does not support VMS, MsDos nor MacOS 9
- does not have FIPS support
- different set of ciphers (-SRP, +ChaCha, +poly1305)
- BIO_* functions do the right thing™
- malloc(x*y) has been converted to reallocarray(x,y)

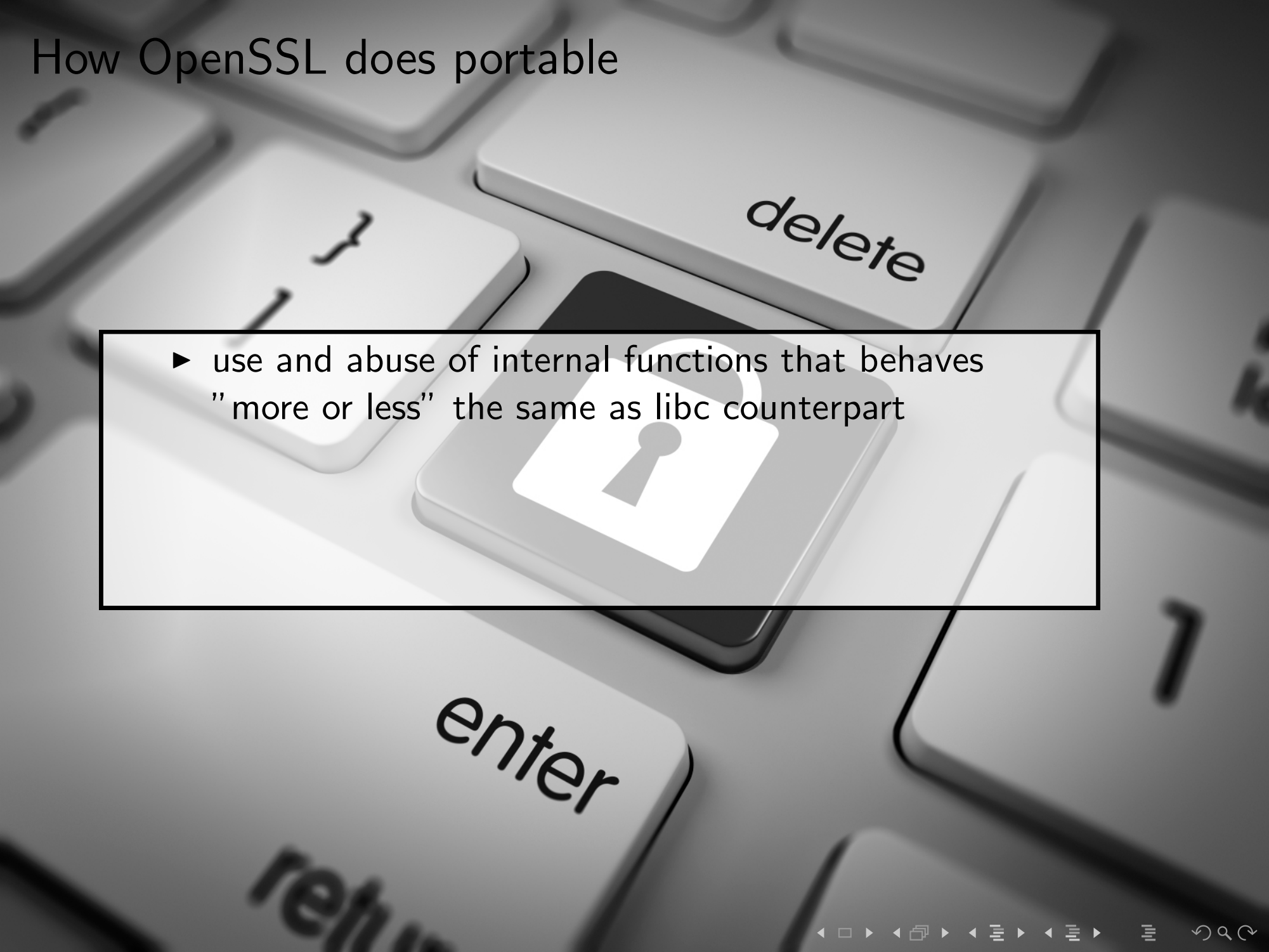
# Some differences between LibreSSL and OpenSSL

- ∼90000 lines of code less but same functionalities
- does not support VMS, MsDos nor MacOS 9
- does not have FIPS support
- different set of ciphers (-SRP, +ChaCha, +poly1305)
- BIO_* functions do the right thing™
- malloc(x*y) has been converted to reallocarray(x,y)
- in LibreSSL, openssl(1) does not have SSLv3 support nor dynamic engine loading support

# Some differences between LibreSSL and OpenSSL

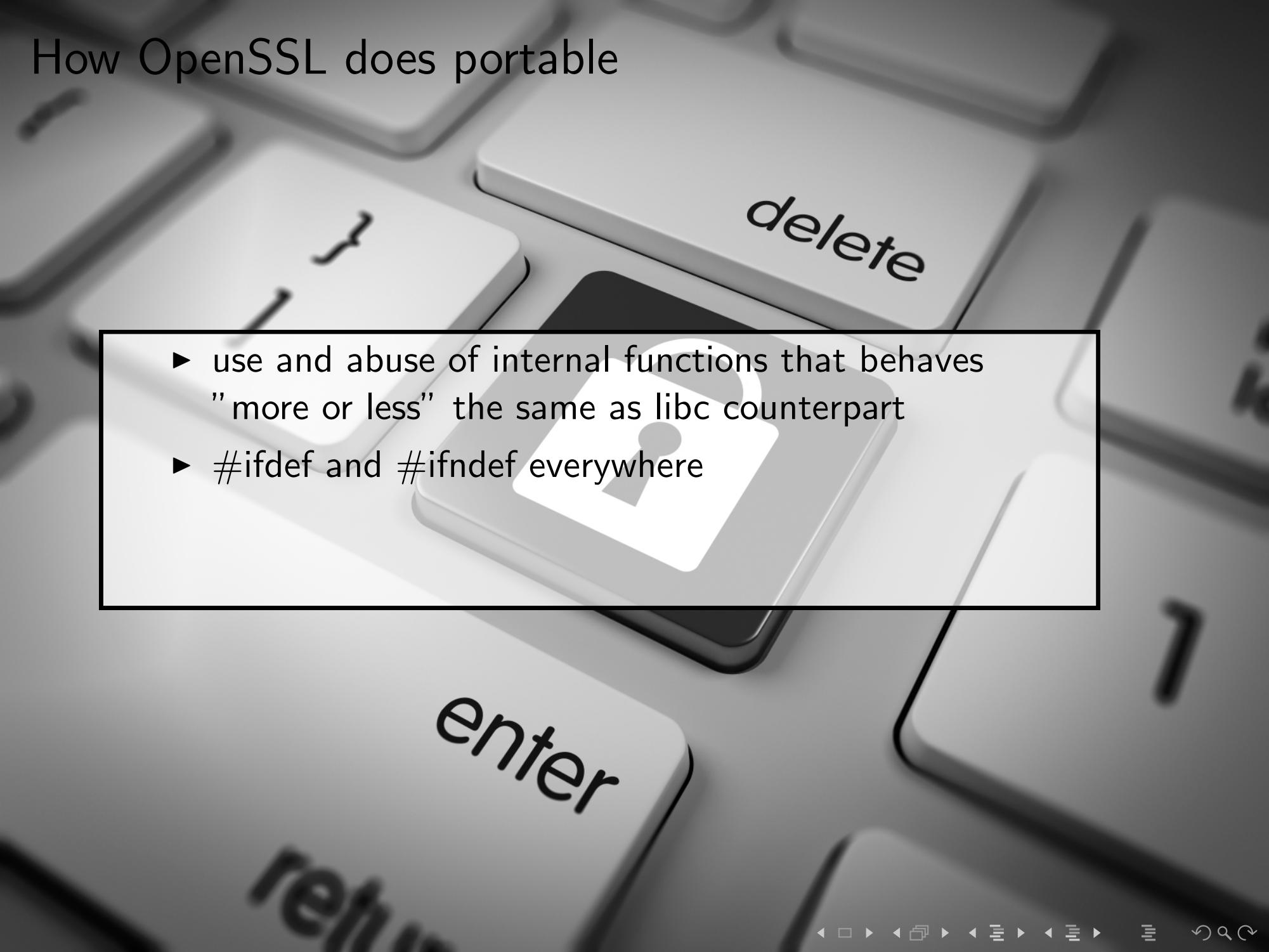
- ~90000 lines of code less but same functionalities
- does not support VMS, MsDos nor MacOS 9
- does not have FIPS support
- different set of ciphers (-SRP, +ChaCha, +poly1305)
- BIO_* functions do the right thing™
- malloc(x*y) has been converted to reallocarray(x,y)
- in LibreSSL, openssl(1) does not have SSLv3 support nor dynamic engine loading support
- OpenSSL can pass environment variables to $ENV:: in config files

# How OpenSSL does portable



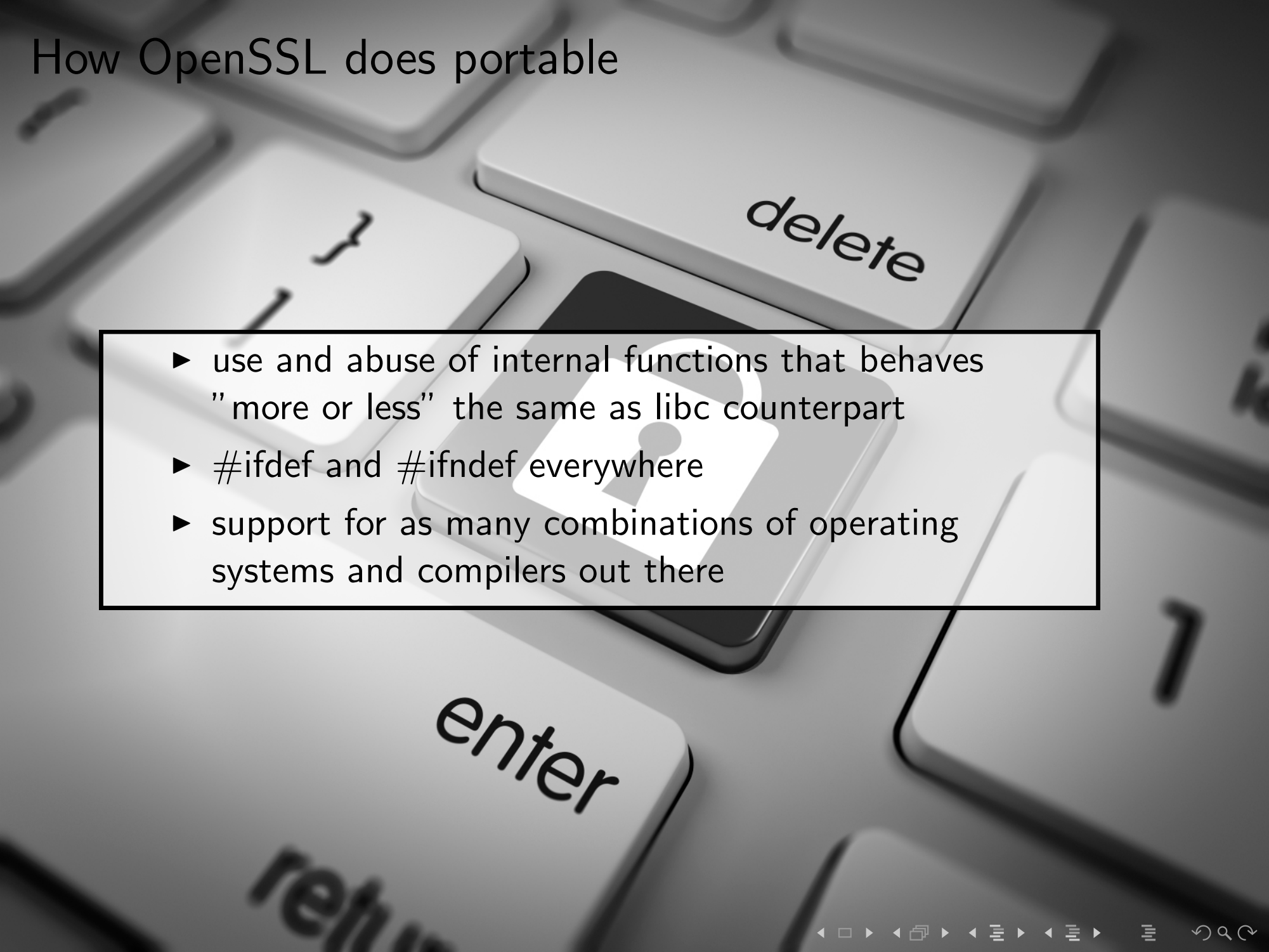

- use and abuse of internal functions that behaves "more or less" the same as libc counterpart

# How OpenSSL does portable




- use and abuse of internal functions that behaves "more or less" the same as libc counterpart
- #ifdef and #ifndef everywhere

# How OpenSSL does portable

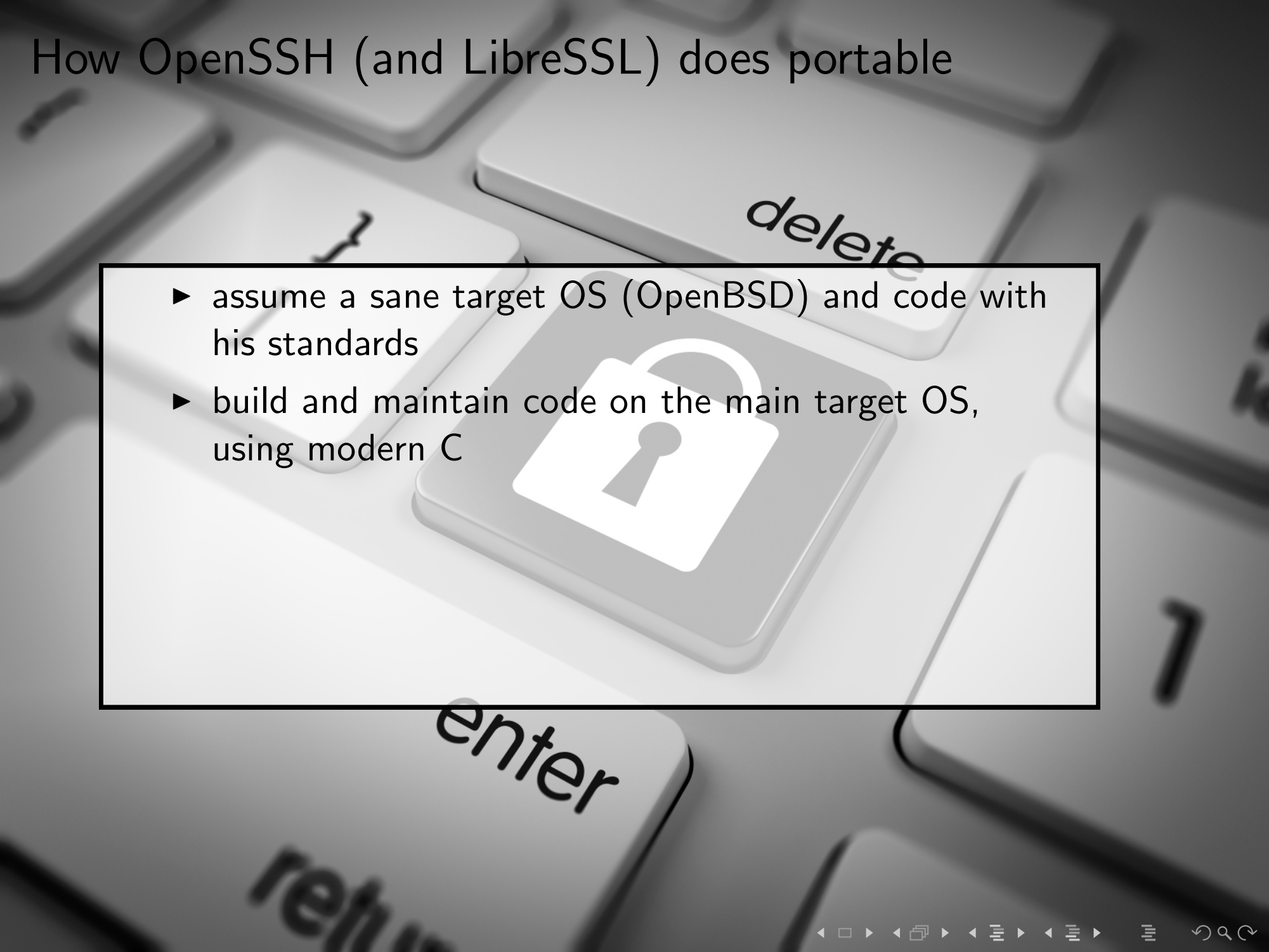



- use and abuse of internal functions that behaves "more or less" the same as libc counterpart
- #ifdef and #ifndef everywhere
- support for as many combinations of operating systems and compilers out there

# How OpenSSH (and LibreSSL) does portable

- assume a sane target OS (OpenBSD) and code with his standards

# How OpenSSH (and LibreSSL) does portable




- assume a sane target OS (OpenBSD) and code with his standards
- build and maintain code on the main target OS, using modern C

# How OpenSSH (and LibreSSL) does portable

- assume a sane target OS (OpenBSD) and code with his standards
- build and maintain code on the main target OS, using modern C
- provide portability code only to provide functions that other OS's don't provide

# How OpenSSH (and LibreSSL) does portable

- assume a sane target OS (OpenBSD) and code with his standards
- build and maintain code on the main target OS, using modern C
- provide portability code only to provide functions that other OS's don't provide
- do not reimplement libc

# How OpenSSH (and LibreSSL) does portable

- assume a sane target OS (OpenBSD) and code with his standards
- build and maintain code on the main target OS, using modern C
- provide portability code only to provide functions that other OS's don't provide
- do not reimplement libc
- put as few #ifdefs as possible in the code

# LibreSSL API, ftp client

```
-               if (inet_pton(AF_INET,  host, &addrbuf) != 1 &&
-                   inet_pton(AF_INET6, host, &addrbuf) != 1) {
-                       if (SSL_set_tlsext_host_name(ssl, host) == 0) {
-                               ERR_print_errors_fp(ttyout);
-                               goto cleanup_url_get;
-                       }
-               }
-               if (SSL_connect(ssl) <= 0) {
-                       ERR_print_errors_fp(ttyout);
+               if (tls_connect_socket(tls, s, sslhost) != 0) {
+                       fprintf(ttyout, "SSL failure: %s\n", tls_error(tls));
                        goto cleanup_url_get;
                }
-               if (ssl_verify) {
-                       X509    *cert;
-
-                       cert = SSL_get_peer_certificate(ssl);
-                       if (cert == NULL) {
-                               fprintf(ttyout, "%s: no server certificate\n",
-                                   getprogname());
-                               goto cleanup_url_get;
-                       }
-
-                       if (ssl_check_hostname(cert, host) != 0) {
-                               X509_free(cert);
-                               fprintf(ttyout, "%s: host `%s' not present in"
-                                   " server certificate\n",
-                                   getprogname(), host);
-                               goto cleanup_url_get;
-                       }
-
-                       X509_free(cert);
```

# Questions ?


Giovanni Bechis
<giovanni@openbsd.org>

SNB Information Technology & Web Solutions