Predictable PRNG In The Vulnerable Debian OpenSSL Package

The What And The How

Luciano Bello[1,2] Maximiliano Bertacchini[2]
luciano at debian.org mbertacchini at citefa.gov.ar

[1]Debian Project

[2]Si6 Labs - CITEFA
(Instituto de Investigaciones Científicas y Técnicas para la Defensa, Argentina)

25th Chaos Communication Congress
Berlin, Germany
December 27-30, 2008

Introduction to DSA-1571

http://www.debian.org/security/2008/dsa-1571

The pseudorandom number generator (PRNG) in Debian's OpenSSL package has been predictable for two years

- Caused by an incorrect Debian-specific patch
- As a result, cryptographic key material is guessable
- Affects other packages linked with `libssl`
- Affects other operating systems which are based on Debian
- Other systems are affected if weak keys (even the public ones) are imported into them
- The first vulnerable version is `0.9.8c-1` (uploaded on Sept 17 2006), propagated into the current stable (Etch)
- The advisory and the patch were released on 13rd May, 2008
- Keys generated with GnuPG or GNUTLS are not affected!

In Other Words

The PRNG output stream is guessable by brute-force, because it's only depends on the process ID that generates it.

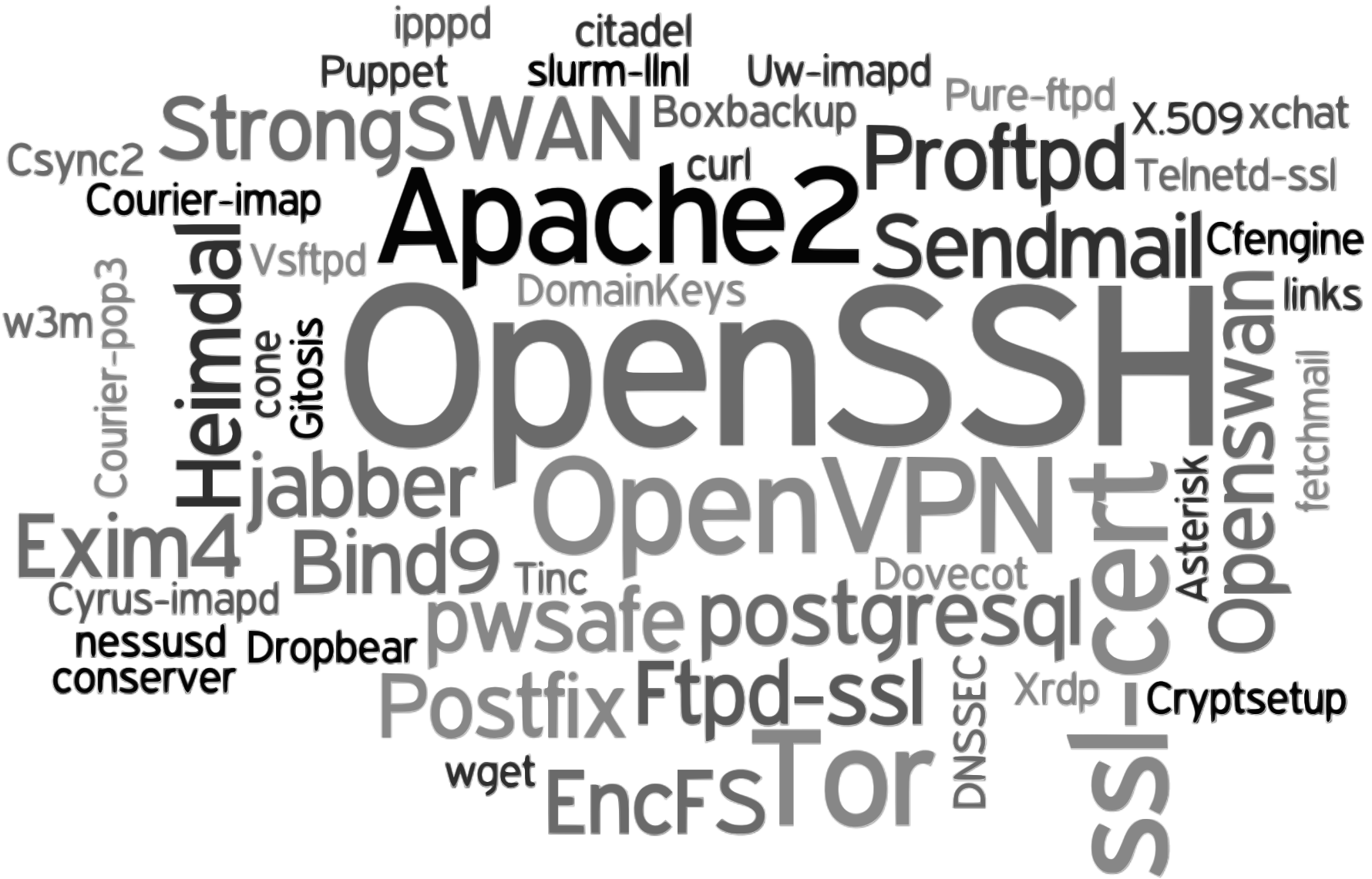
As a consequence the key space for a particular cryptosystem depends on:

- `PID_MAX` (typicaly 2^{15})
- The host architecture / endianness
- Internal state (number of requested random bits, etc)

This affects to every aspect in cryptography:

- Client and server authentication
- DSA signatures
- Key agreement
- Session keys
- Nonsecret random data (Salt, IV)
- Autogenerated passwords/passphrases (like OTP)

Affected Packages



A Bad Beginning

"A series of unfortunate events"

- Richard Kettlewell files "Valgrind-Clean the RNG" on Wed 19 Apr 2006 15:18:15 UTC (`http://bugs.debian.org/363516`)
- Kurt (the Debian OpenSSL maintainer) identified two problematic lines (`http://bugs.debian.org/363516#10`)
  "What it's doing is adding uninitialised numbers to the pool to create random numbers. I've been thinking about commenting those out."

CRYPTO/RAND/MD_RAND.C

```
274:
        MD_Update(&m,buf,j);
467:
#ifndef PURIFY
        MD_Update(&m,buf,j); /* purify complains */
#endif
```

A Bad Beginning
(cont.)

Uninitialized memory can produce non-deterministic behavior in a *normal* program. But sometimes you want non-deterministic behavior, like in a PRNG (though it's not a strong source of entropy)

A Bad Beginning (cont.)

Uninitialized memory can produce non-deterministic behavior in a *normal* program. But sometimes you want non-deterministic behavior, like in a PRNG (though it's not a strong source of entropy)

The Proposed "Solutions"

1. Mark as *wontfix*
   - Gives a lot of fake warnings when using Valgrind
2. Use that Valgrind thing to *tell Valgrind to ignore it*
   - Is a Valgrind specific solution
   - Changes the binary in a strange way
3. *Don't add the buffer* to the pool. All the good entropy comes from `/dev/urandom`.
   - The added entropy is negligible and is there for historical reasons
   - Let's ask in openssl-dev

(`http://www.mail-archive.com/openssl-dev@openssl.org/msg21156.html`)

The Misunderstanding

(http://www.mail-archive.com/openssl-dev@openssl.org/msg21156.html)

Random number generator, uninitialised data and valgrind.

Kurt Roeckx

Mon, 01 May 2006 12:14:11 -0700

```
Hi,

When debbuging applications that make use of openssl using
valgrind, it can show alot of warnings about doing a conditional
jump based on an unitialised value.  Those unitialised values are
generated in the random number generator.  It's adding an
unintialiased buffer to the pool.

The code in question that has the problem are the following 2
pieces of code in crypto/rand/md_rand.c:

247:
                MD_Update(&m,buf,j);

467:
#ifndef PURIFY
                MD_Update(&m,buf,j); /* purify complains */
#endif

Because of the way valgrind works (and has to work), the place
where the unitialised value is first used, and the place were the
```

The Misunderstanding

(http://www.mail-archive.com/openssl-dev@openssl.org/msg21157.html)

Re: Random number generator, uninitialised data and valgrind.

Ulf Möller

Mon, 01 May 2006 15:34:24 -0700

Kurt Roeckx schrieb:

> What I currently see as best option is to actually comment out
> those 2 lines of code. But I have no idea what effect this
> really has on the RNG. The only effect I see is that the pool
> might receive less entropy. But on the other hand, I'm not even
> sure how much entropy some unitialised data has.

Not much. If it helps with debugging, I'm in favor of removing them. (However the last time I checked, valgrind reported thousands of bogus error messages. Has that situation gotten better?)

OpenSSL Project — http://www.openssl.org
Development Mailing List — openssl-dev@openssl.org
Automated List Manager —

The Infamous Patch

Diff for /openssl/trunk/rand/md_rand.c between version 140 and 141

version 140, Tue May 2 16:25:19 2006 UTC | **version 141, Tue May 2 16:34:53 2006 UTC**

Line 271

```
        else
            MD_Update(&m,&(state[st_idx]),j);

        MD_Update(&m,buf,j);
        MD_Update(&m,(unsigned char *)&(md_c[0]),sizeof(md_c));
        MD_Final(&m,local_md);
        md_c[1]++;
```

Line 271

```
        else
            MD_Update(&m,&(state[st_idx]),j);

        /*
         * Don't add uninitialised data.
         */
        MD_Update(&m,buf,j);
        MD_Update(&m,(unsigned char *)&(md_c[0]),sizeof(md_c));
        MD_Final(&m,local_md);
        md_c[1]++;
```

Line 465

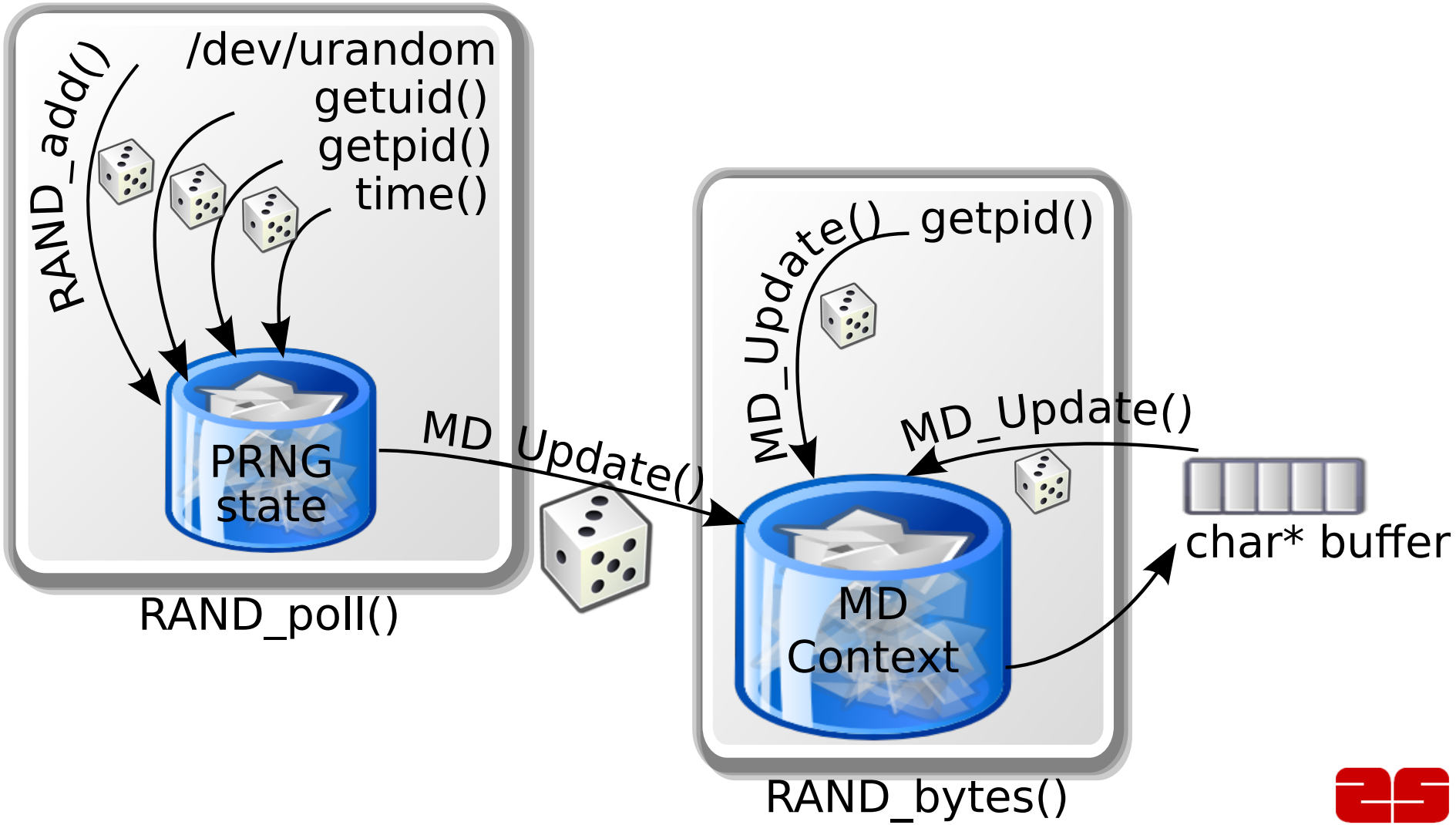
```
        MD_Update(&m,local_md,MD_DIGEST_LENGTH);
        MD_Update(&m,(unsigned char *)&(md_c[0]),sizeof(md_c));
#ifndef PURIFY

        MD_Update(&m,buf,j); /* purify complains */
#endif
        k=(st_idx+MD_DIGEST_LENGTH/2)-st_num;
        if (k > 0)
```

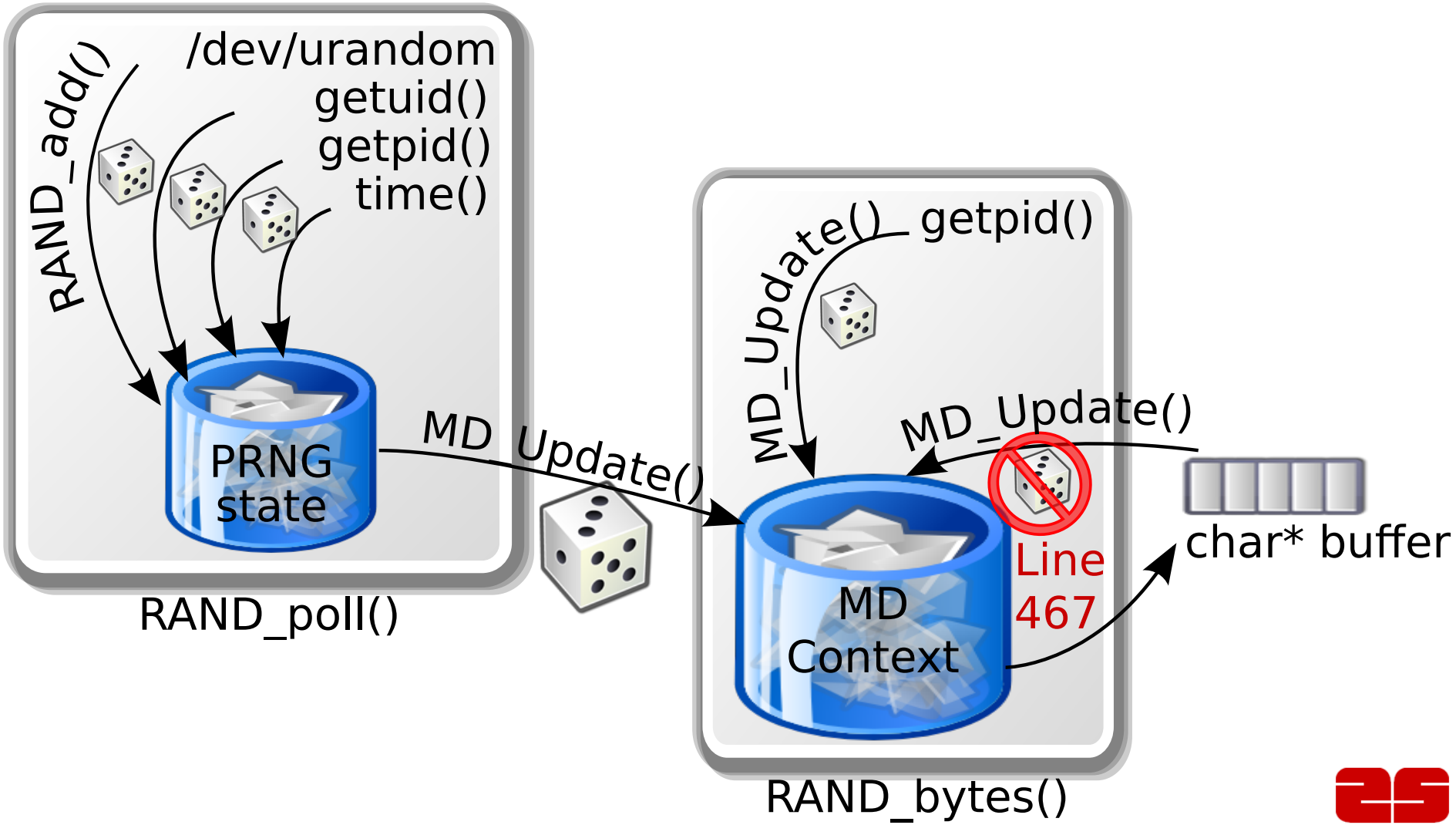
Line 468

```
        MD_Update(&m,local_md,MD_DIGEST_LENGTH);
        MD_Update(&m,(unsigned char *)&(md_c[0]),sizeof(md_c));
#ifndef PURIFY
        /*
         * Don't add uninitialised data.
         */
        MD_Update(&m,buf,j); /* purify complains */
        */
#endif
        k=(st_idx+MD_DIGEST_LENGTH/2)-st_num;
        if (k > 0)
```

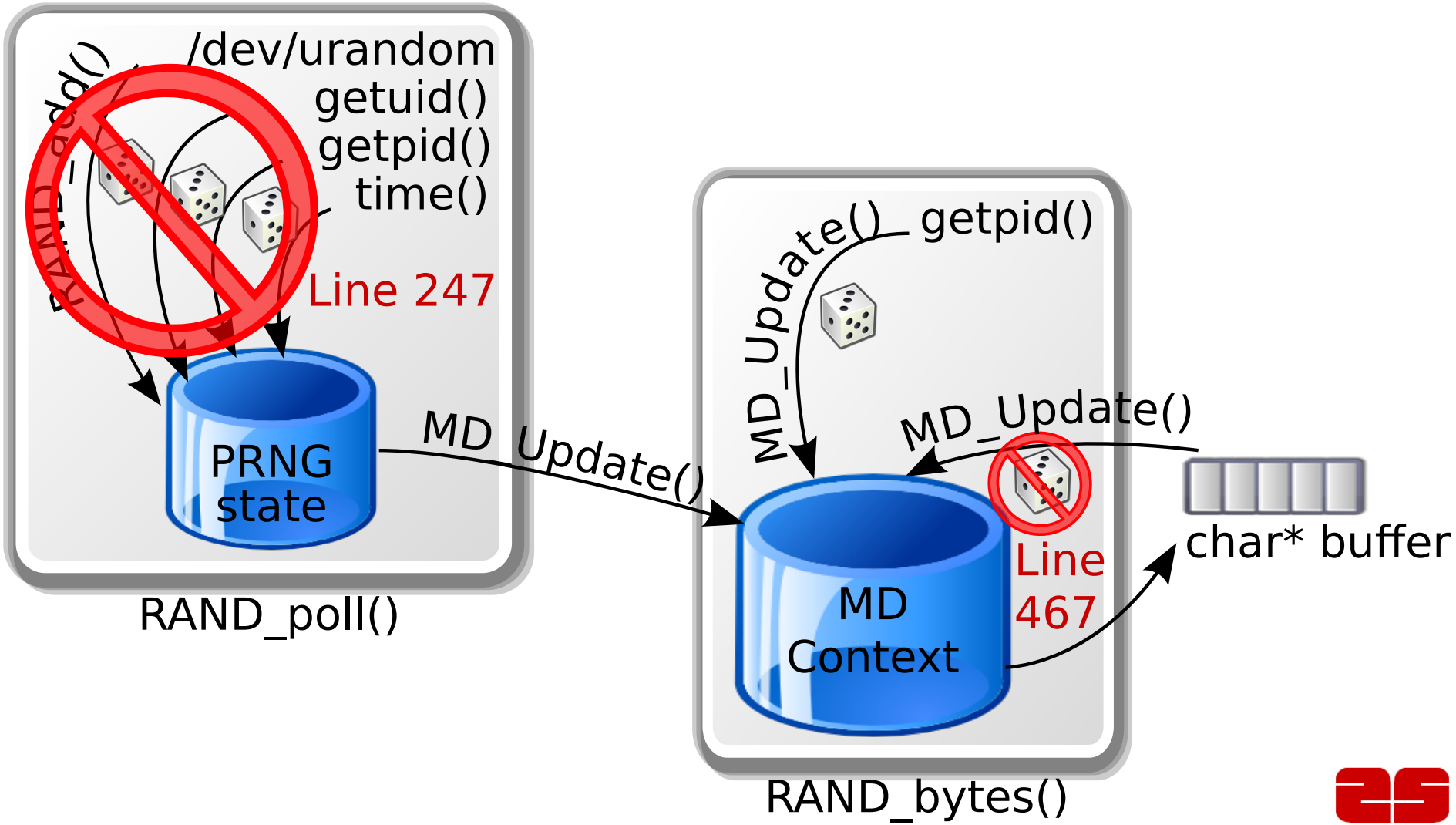
OpenSSL's NRNG



OpenSSL's NRNG



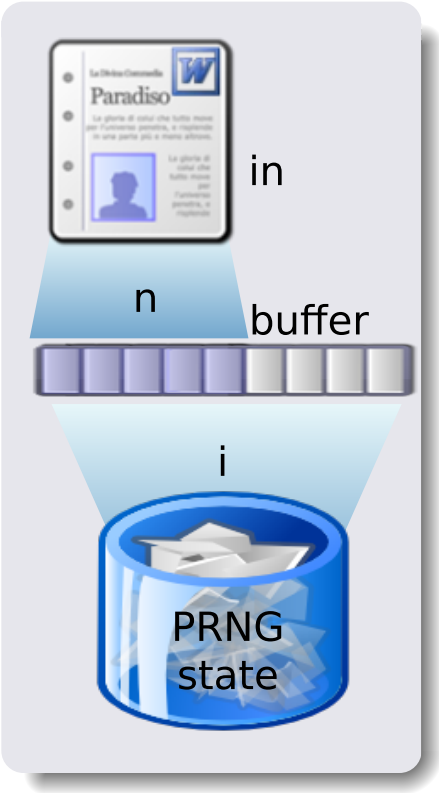
OpenSSL's NRNG



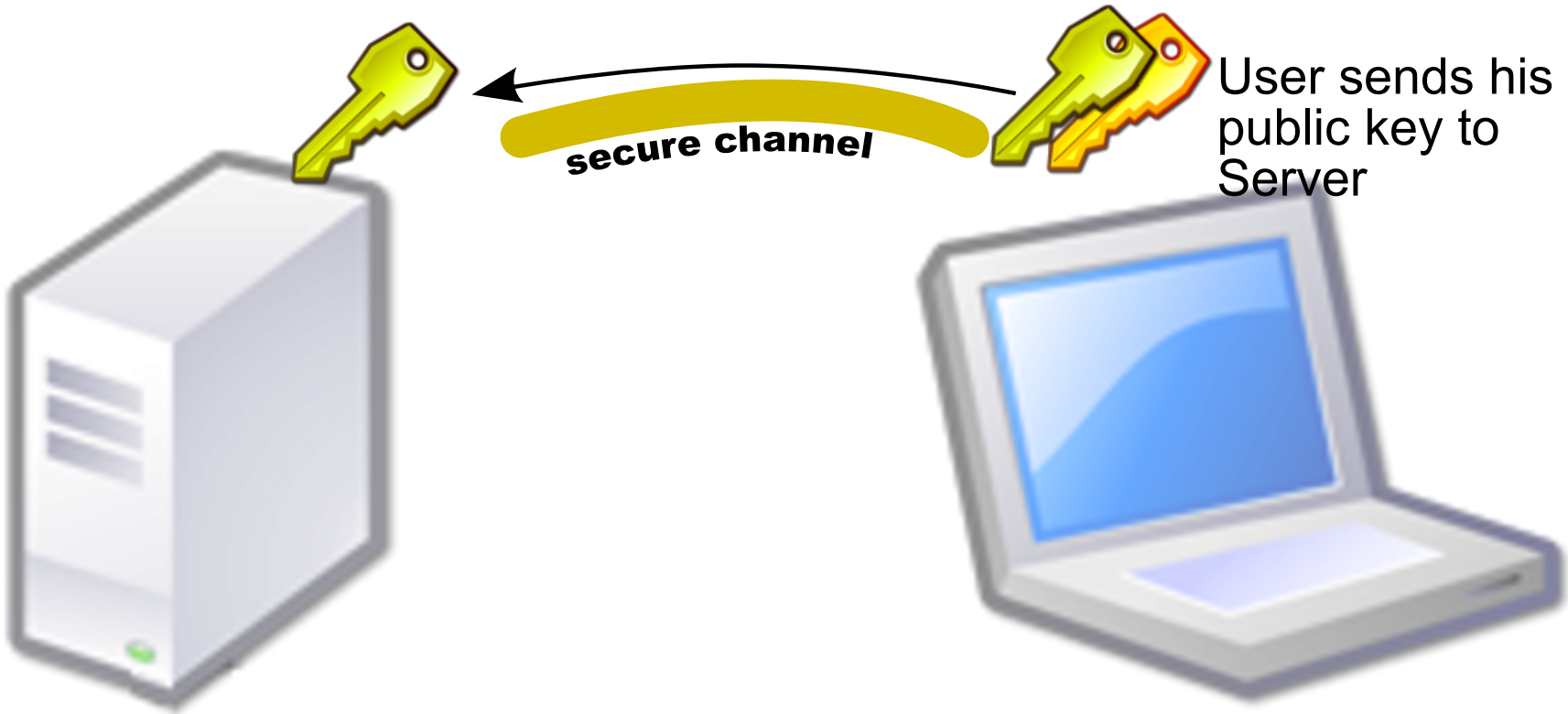
Why had 247 been commented?

Example (OpenSSL API)

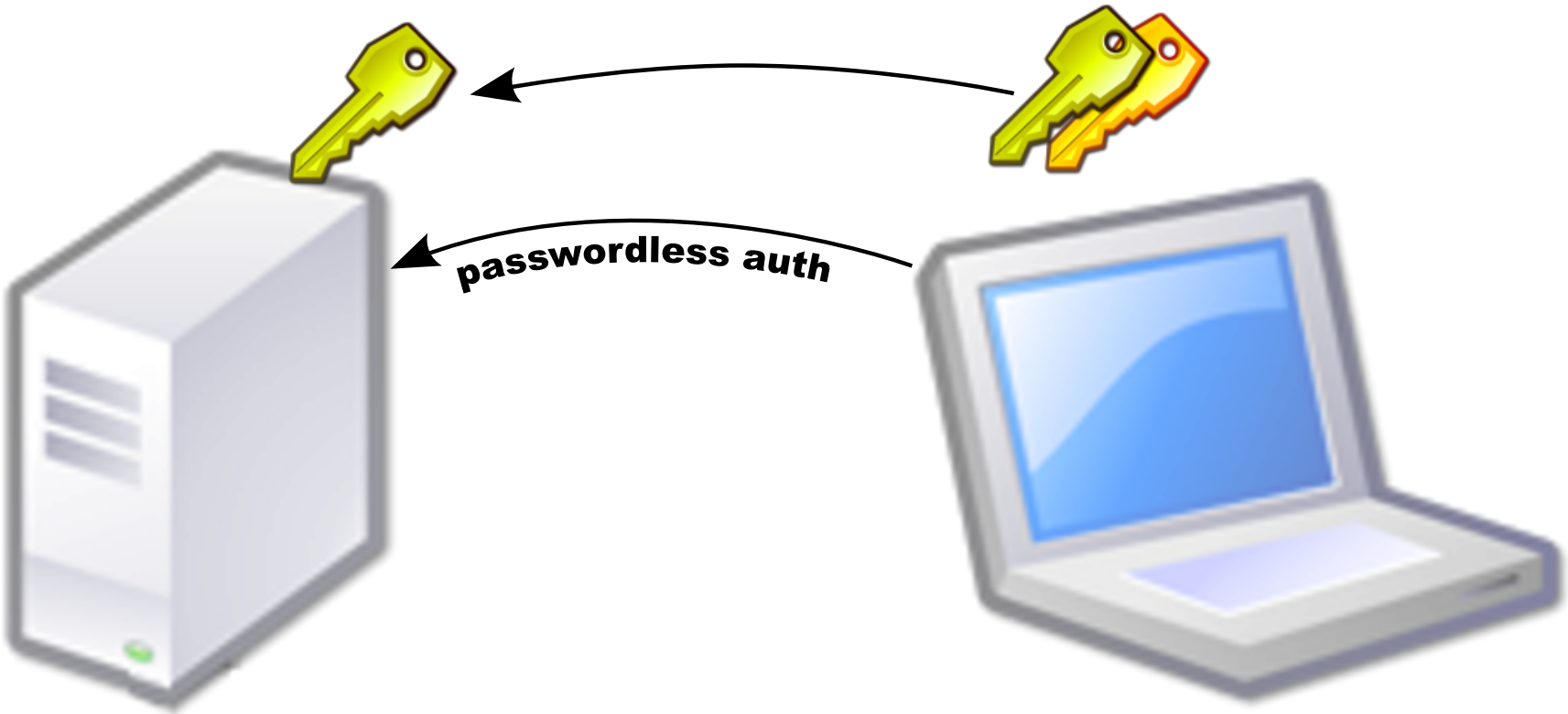
```
int RAND_load_file(const char *file, long bytes)
    {
        ...
        i=fread(buf,1,n,in);
        if (i <= 0) break;
        /* even if n != i, use the full array */
        RAND_add(buf,n,(double)i);
        ...
    }
```



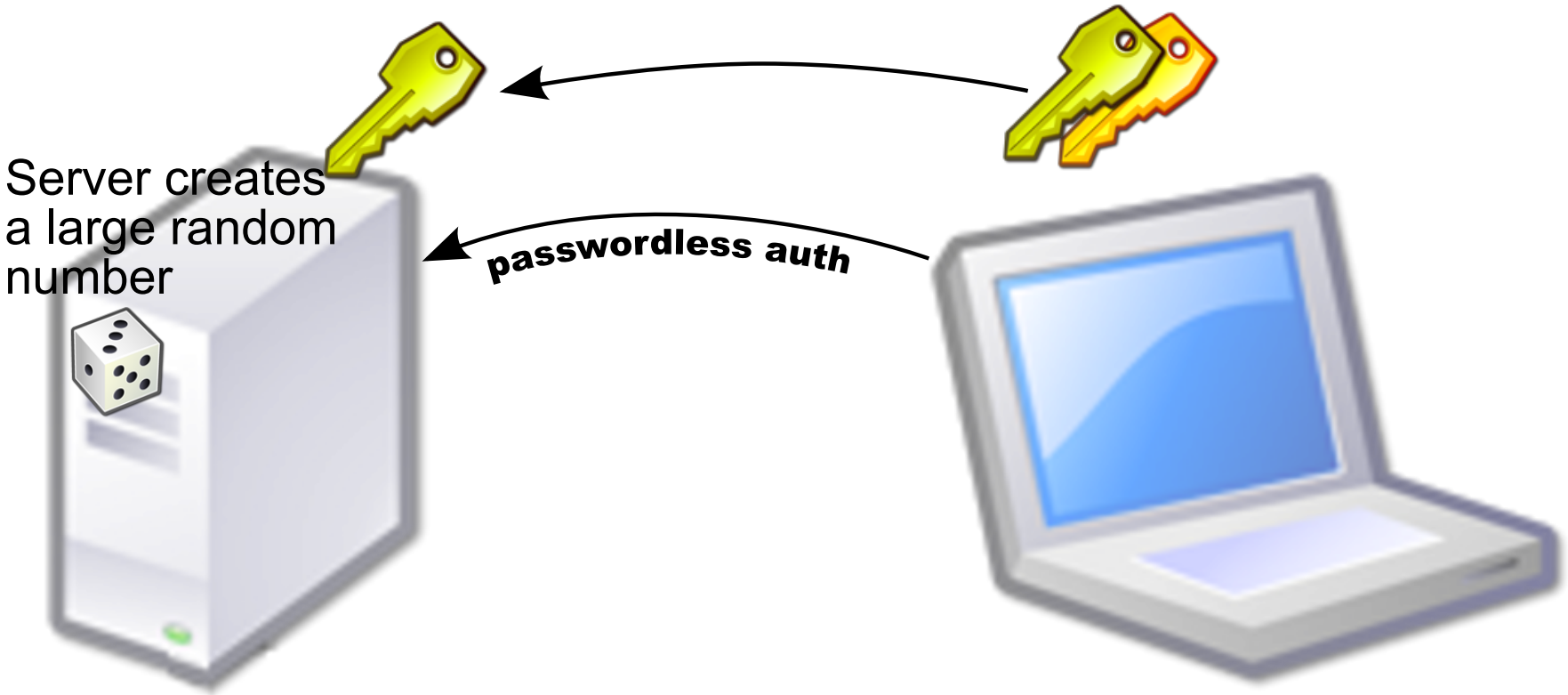
Challenge Authentication



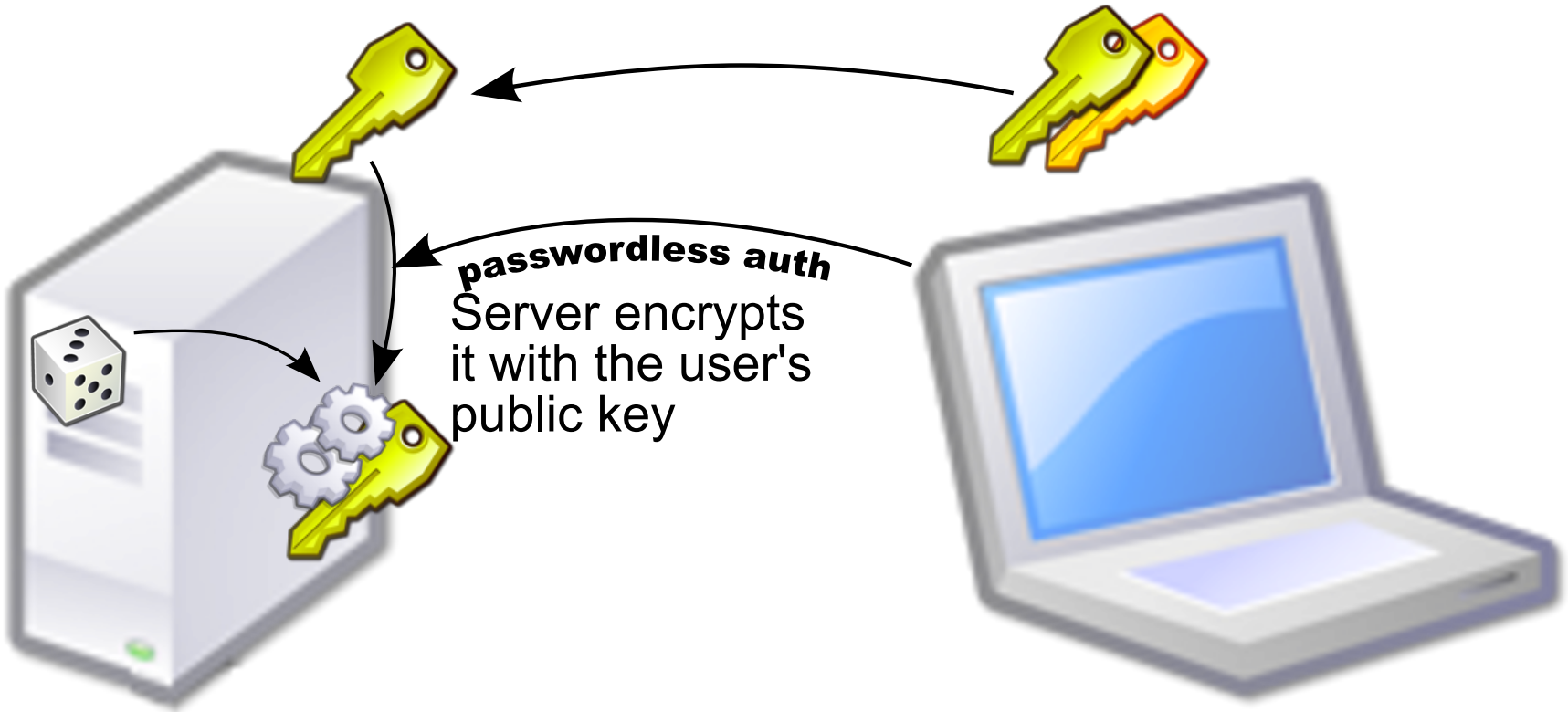
Challenge Authentication



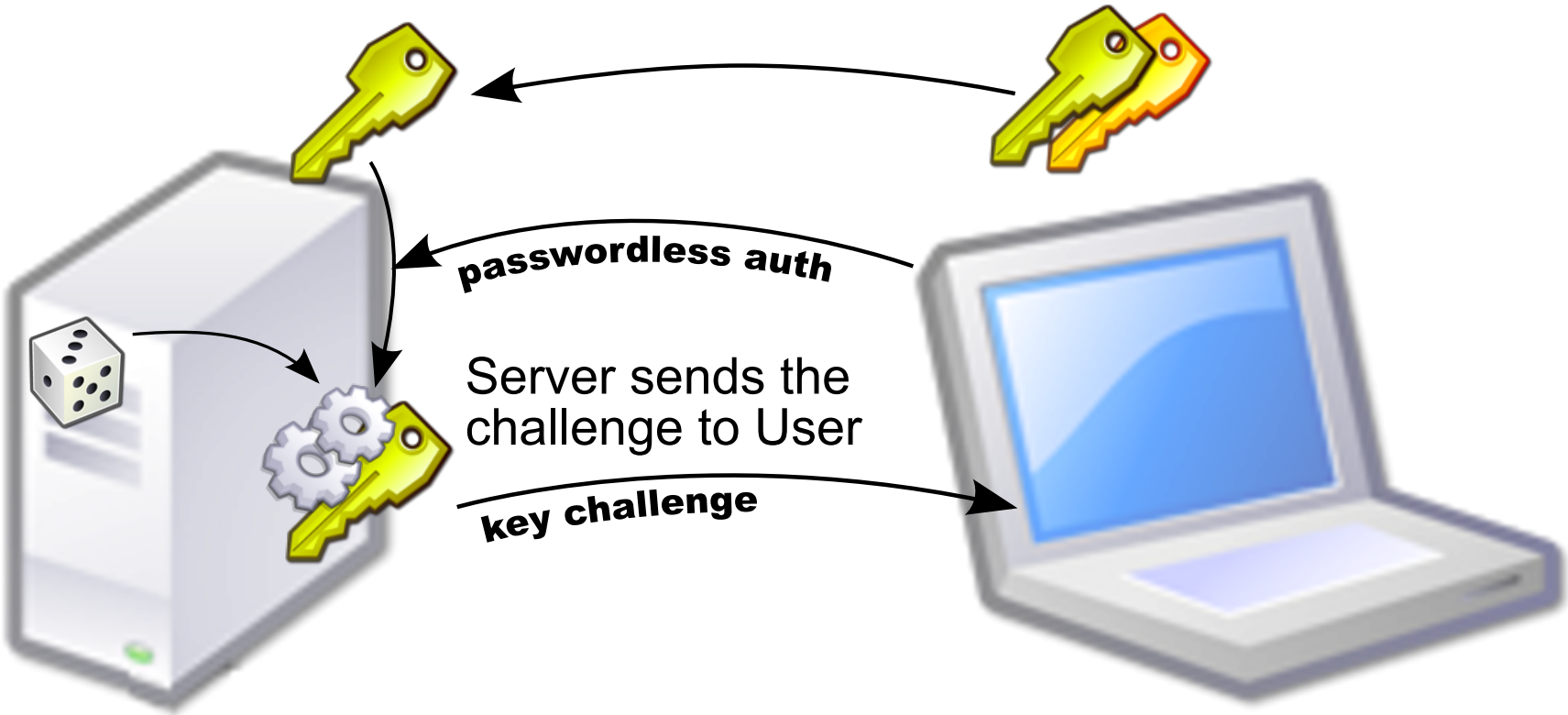
Challenge Authentication



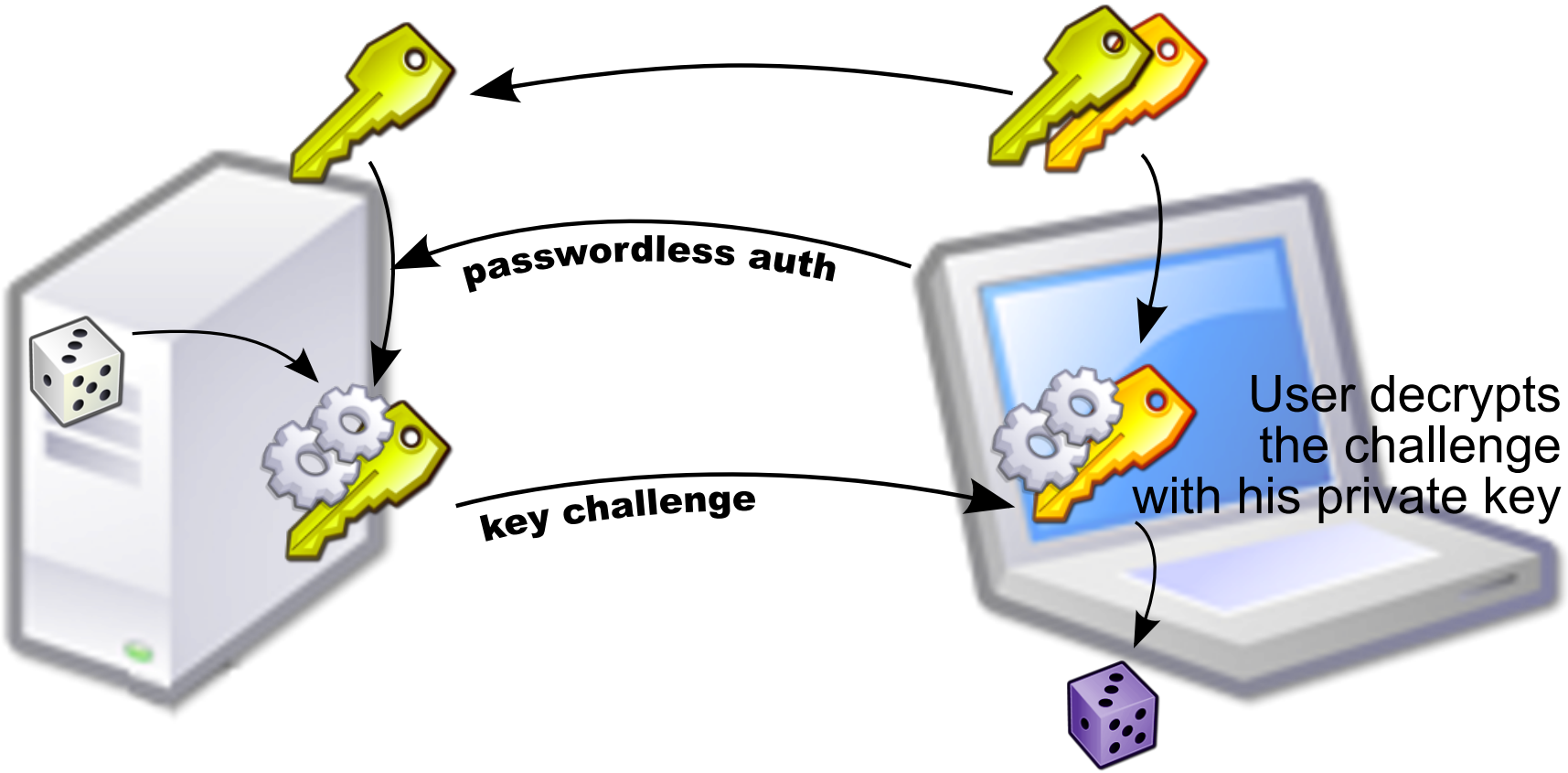
Challenge Authentication



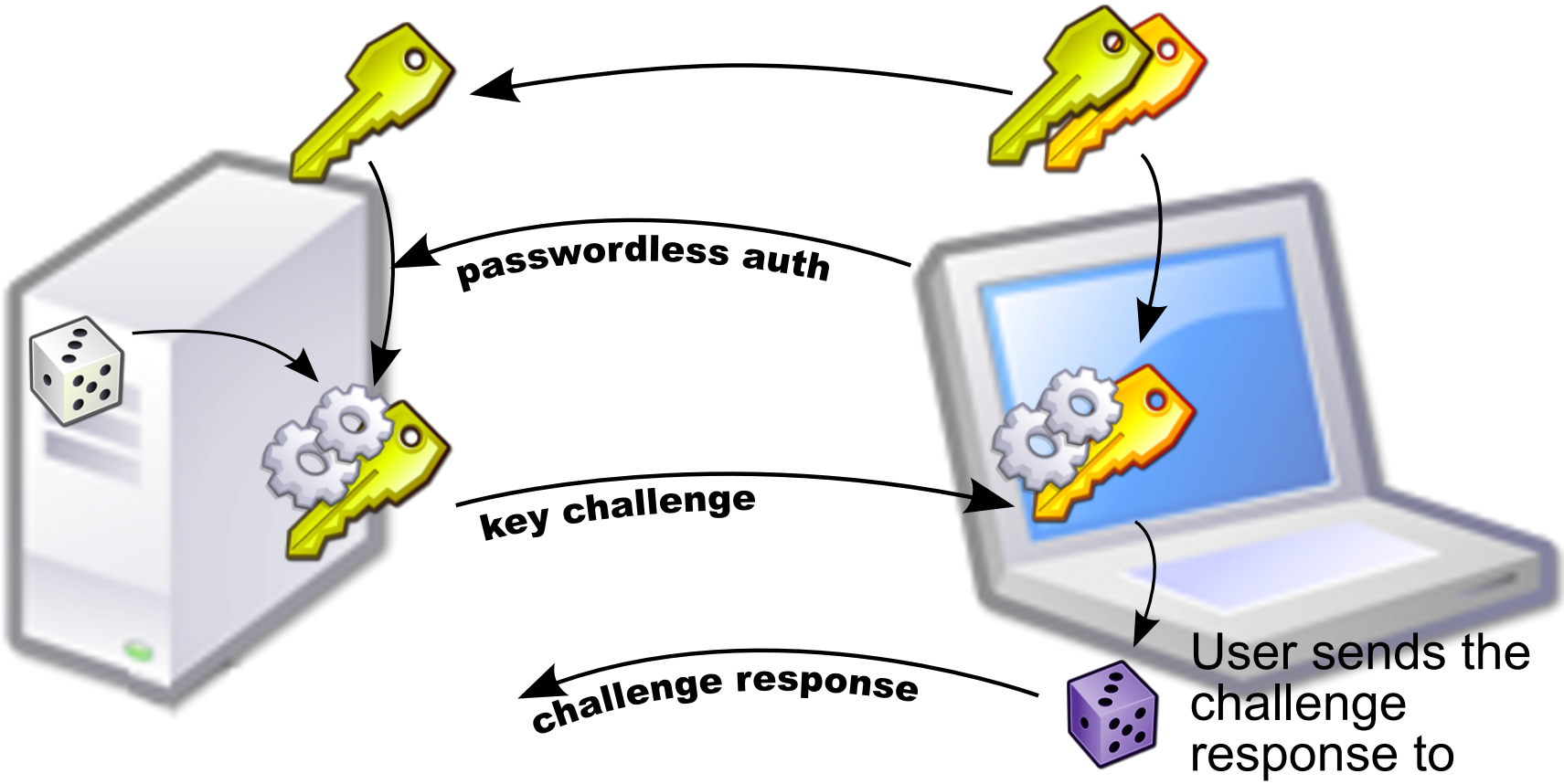
Challenge Authentication



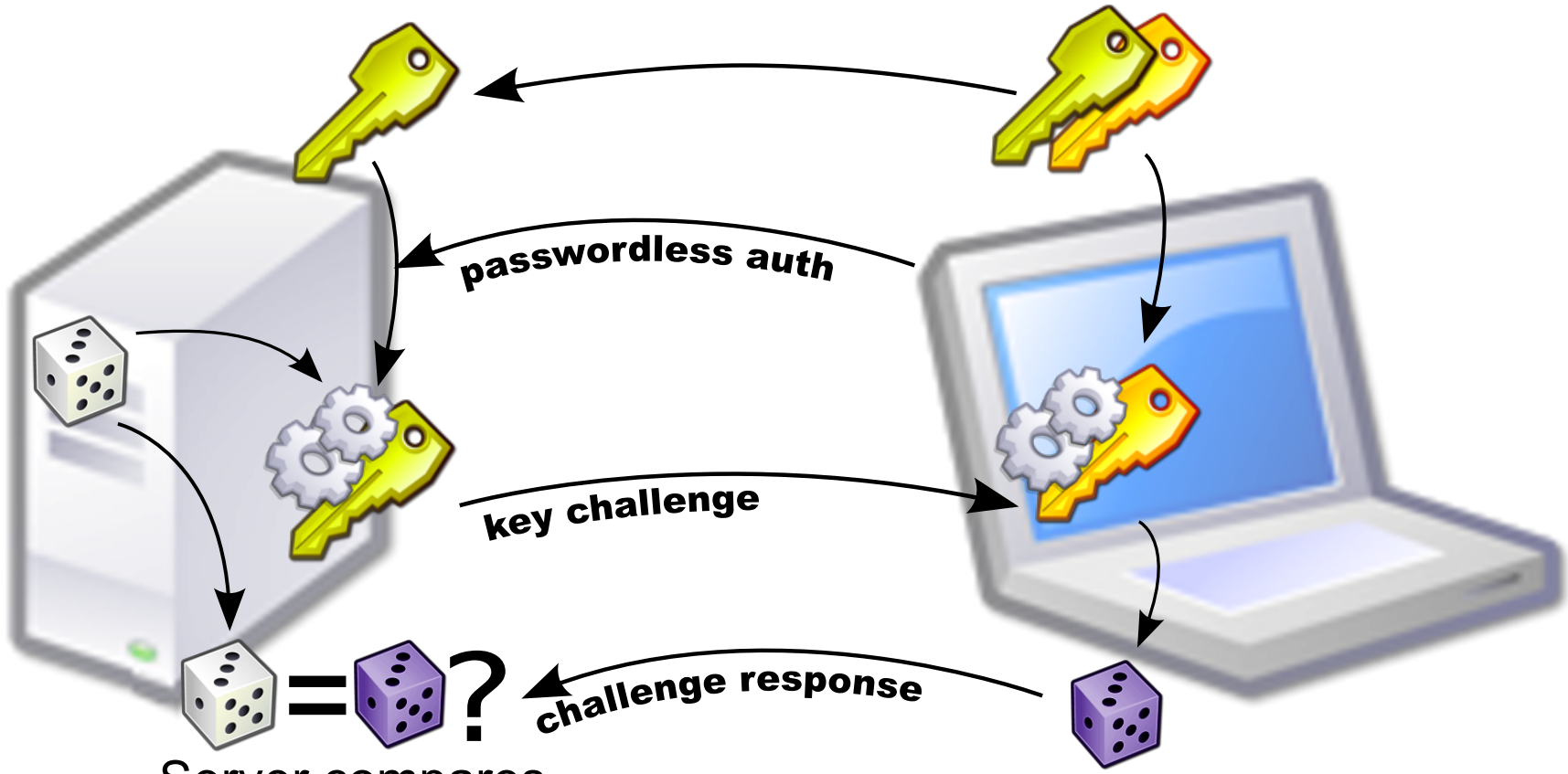
Challenge Authentication



Challenge Authentication



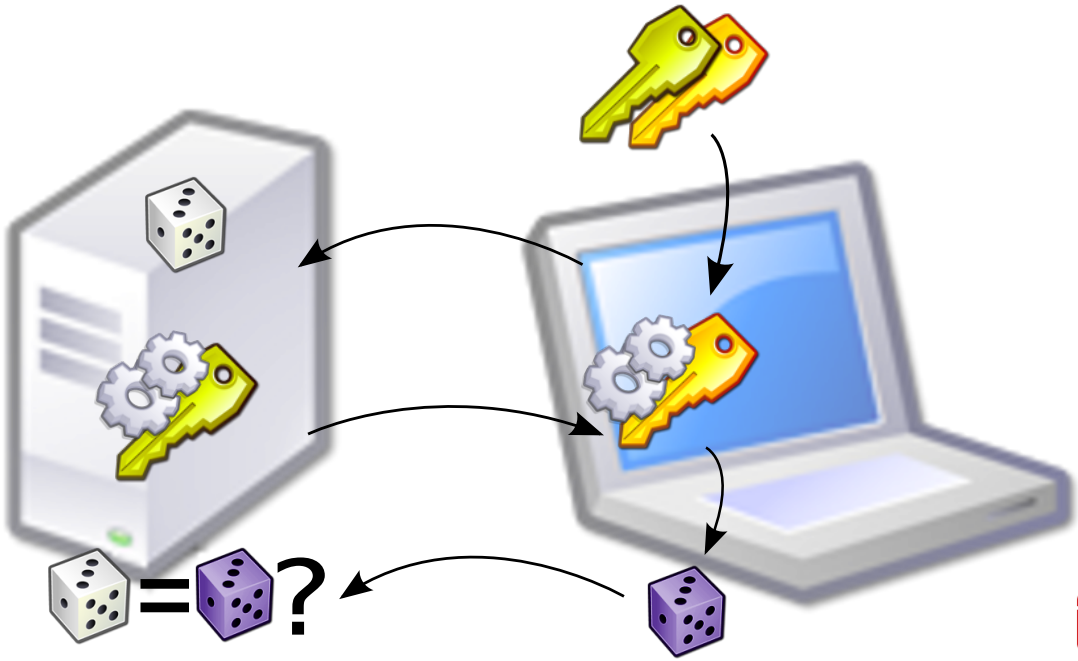
Challenge Authentication



Server compares
If they match access is granted

Challenge Authentication

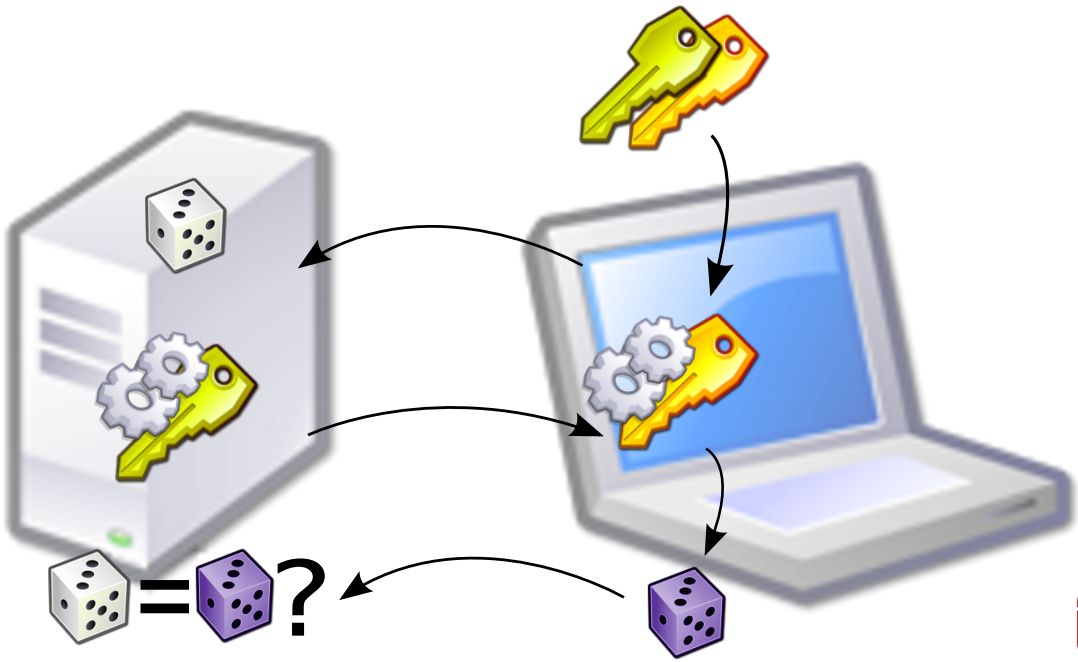
Brute-Force Attack



Challenge Authentication

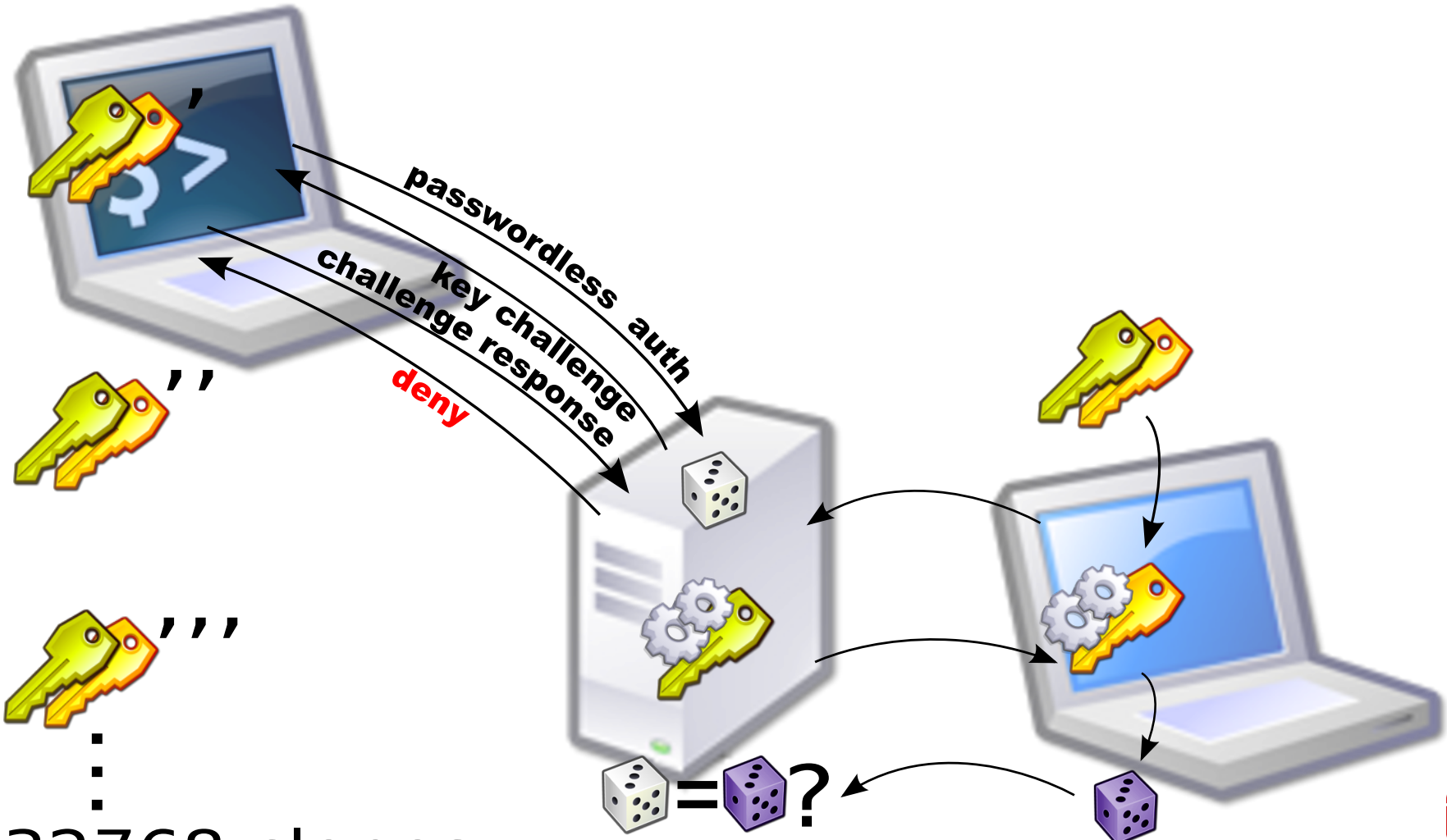
Brute-Force Attack

32768 clones



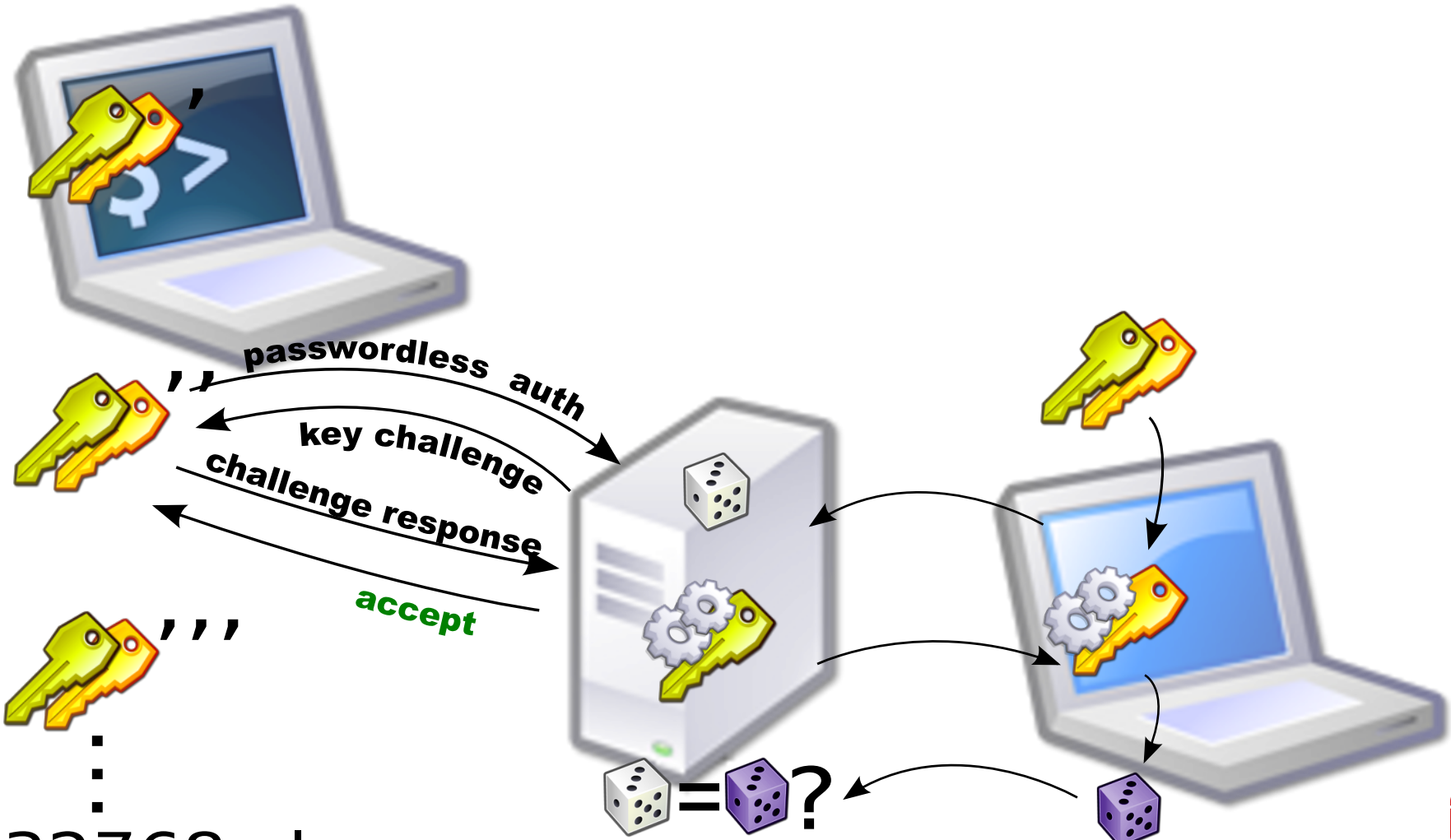
Challenge Authentication

Brute-Force Attack



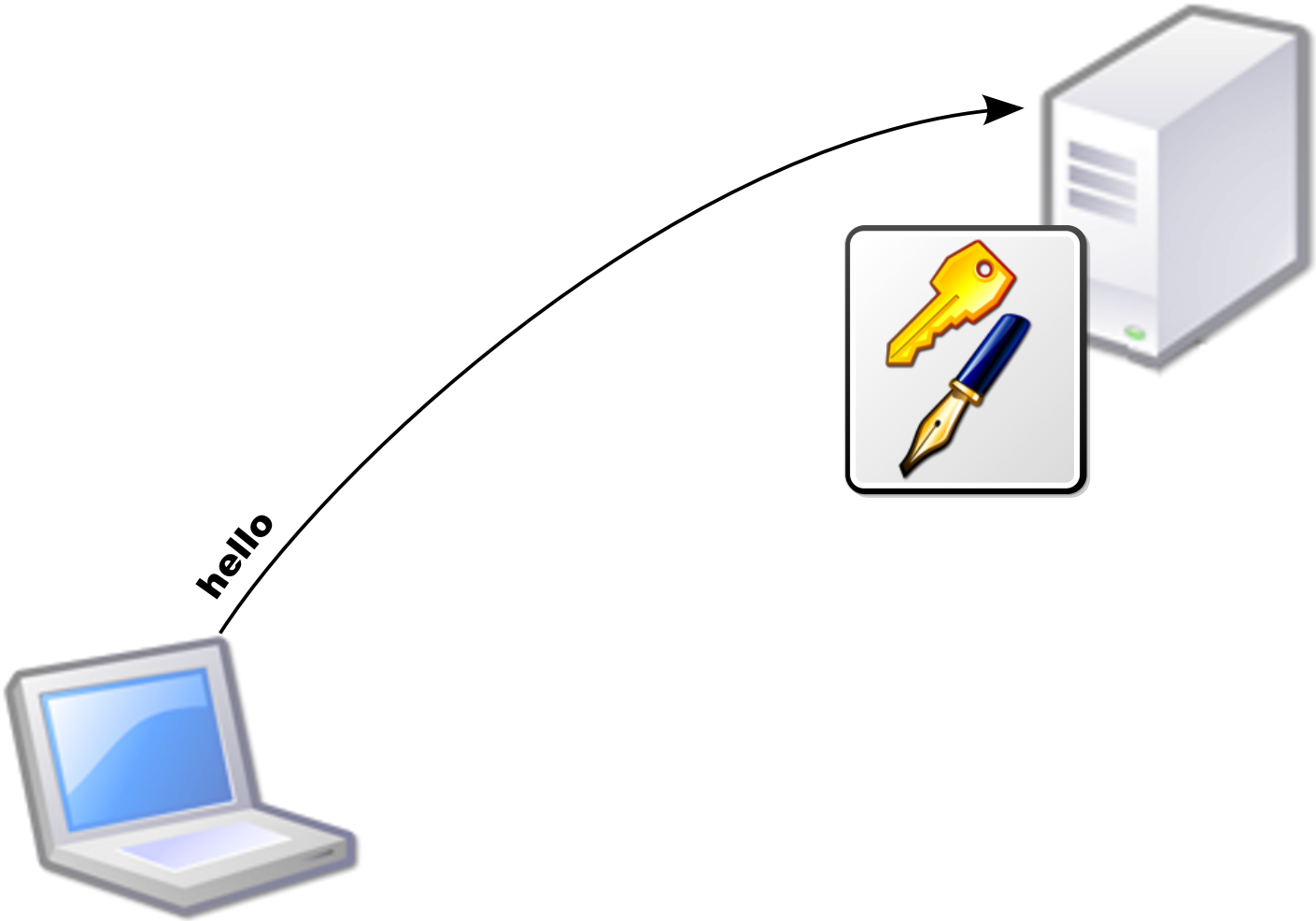
Challenge Authentication

Brute-Force Attack



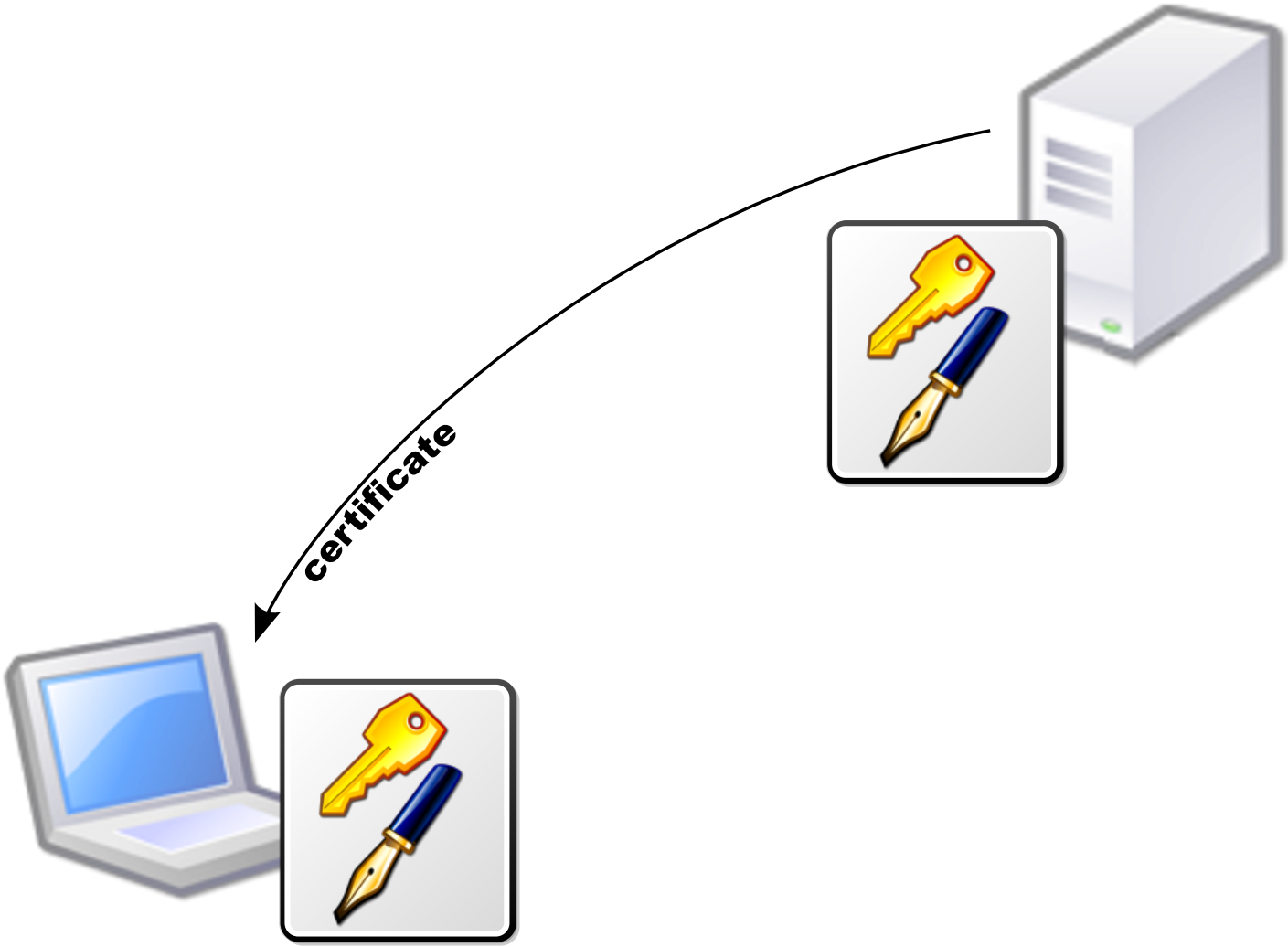
Certificate Authentication

Certificate Attack (MitM)



Certificate Authentication

Certificate Attack (MitM)

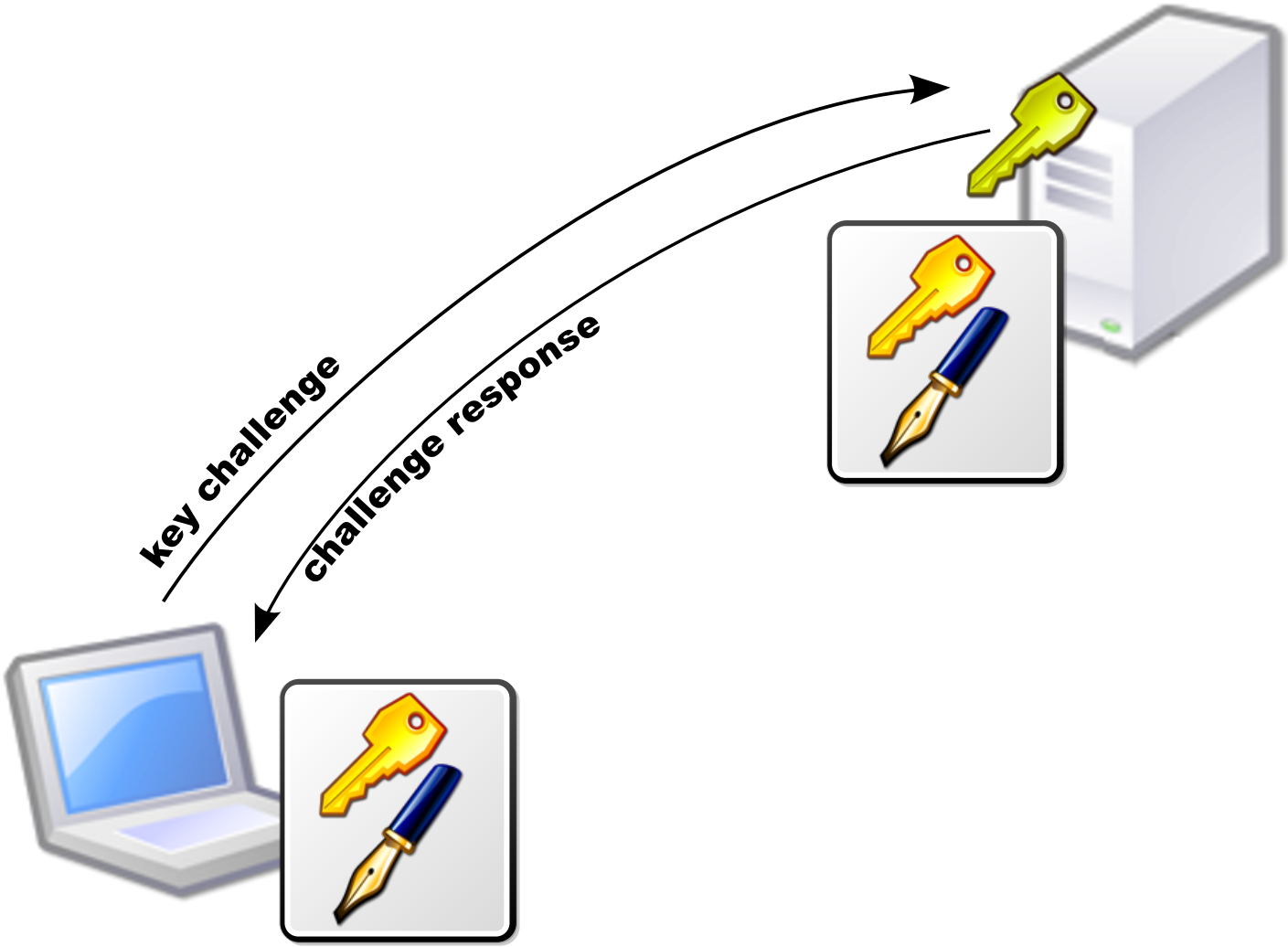


Certificate Authentication

Certificate Attack (MitM)

Certificate Authentication

Certificate Attack (MitM)

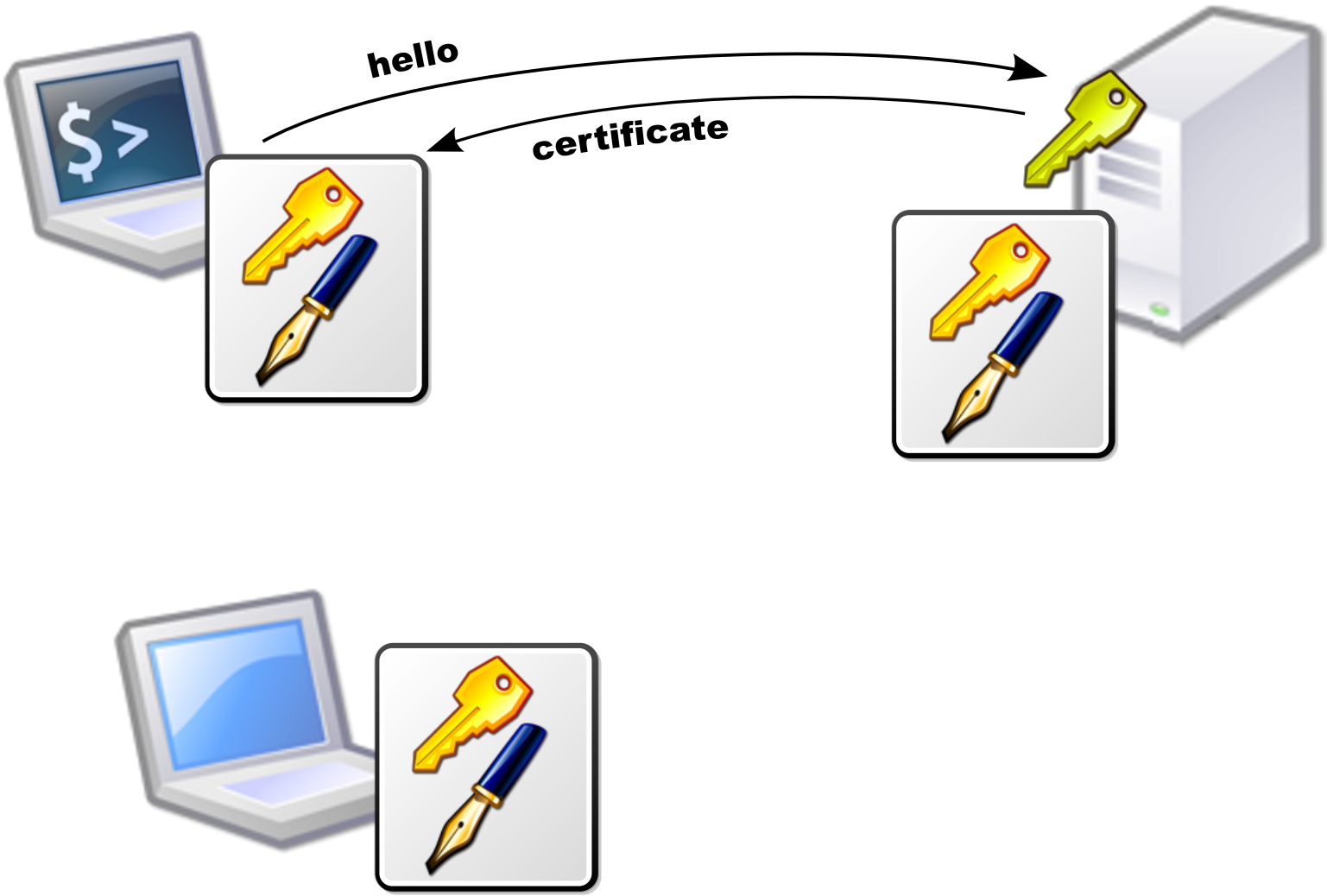


Certificate Authentication

Certificate Attack (MitM)

Certificate Authentication

Certificate Attack (MitM)

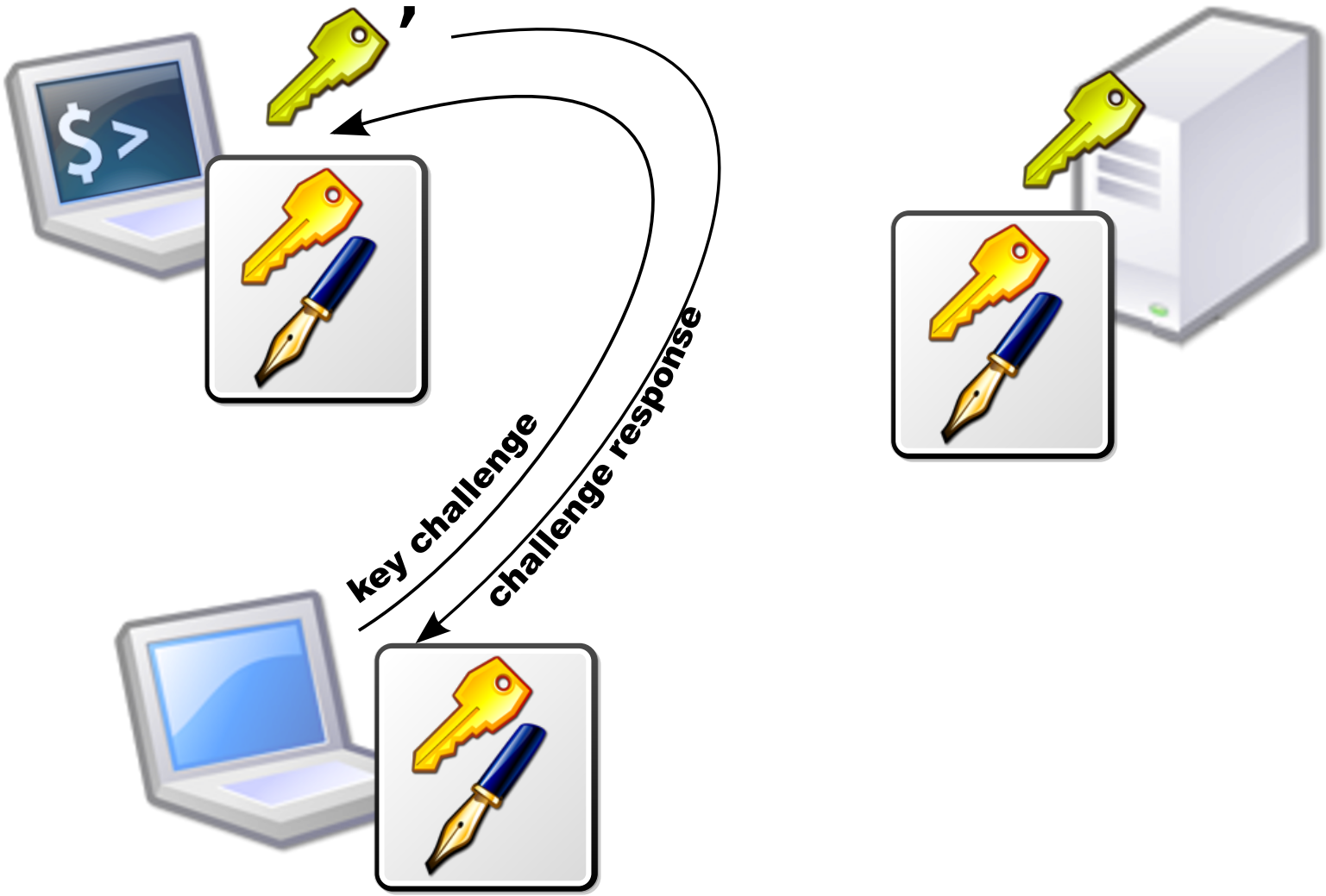


Certificate Authentication

Certificate Attack (MitM)

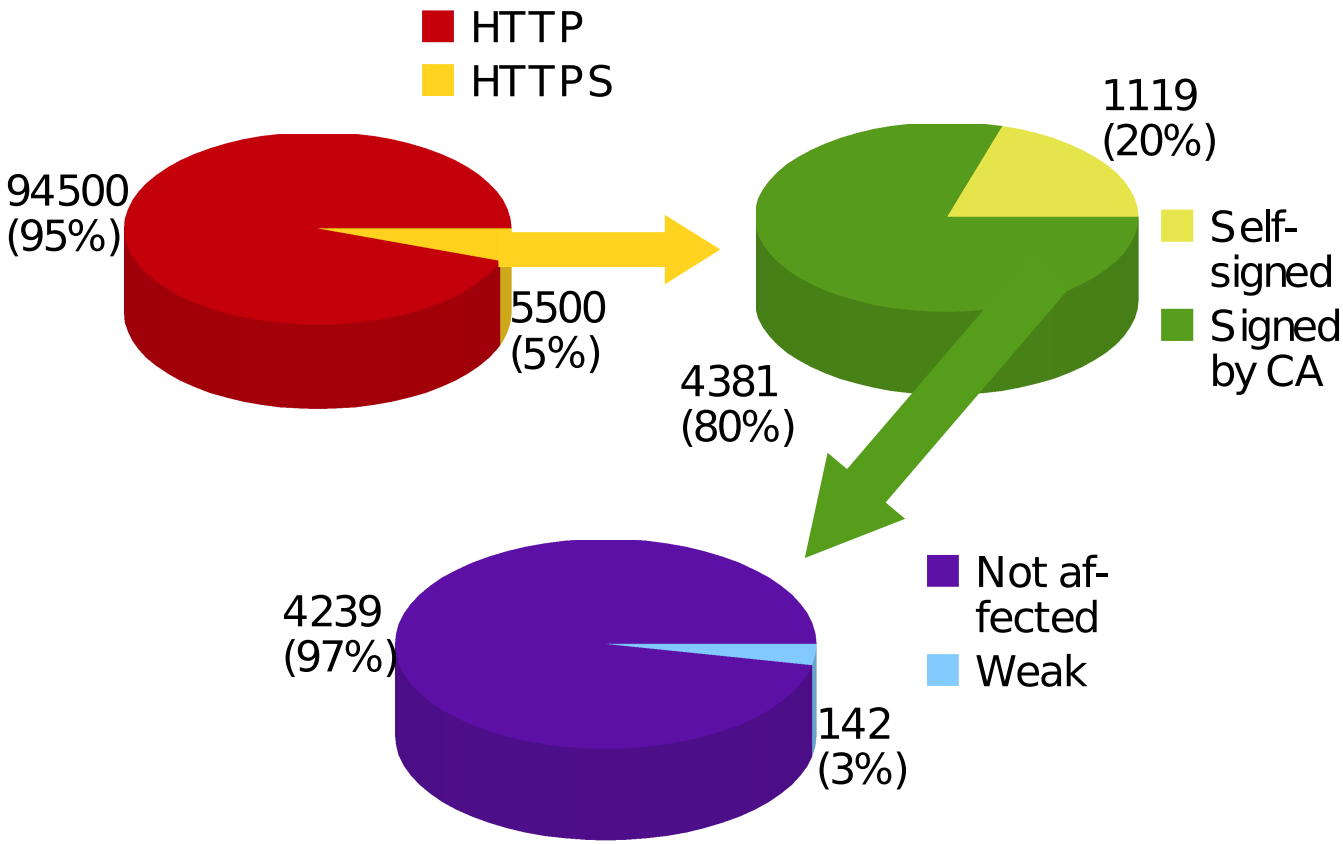
Certificate Authentication

Certificate Attack (MitM)



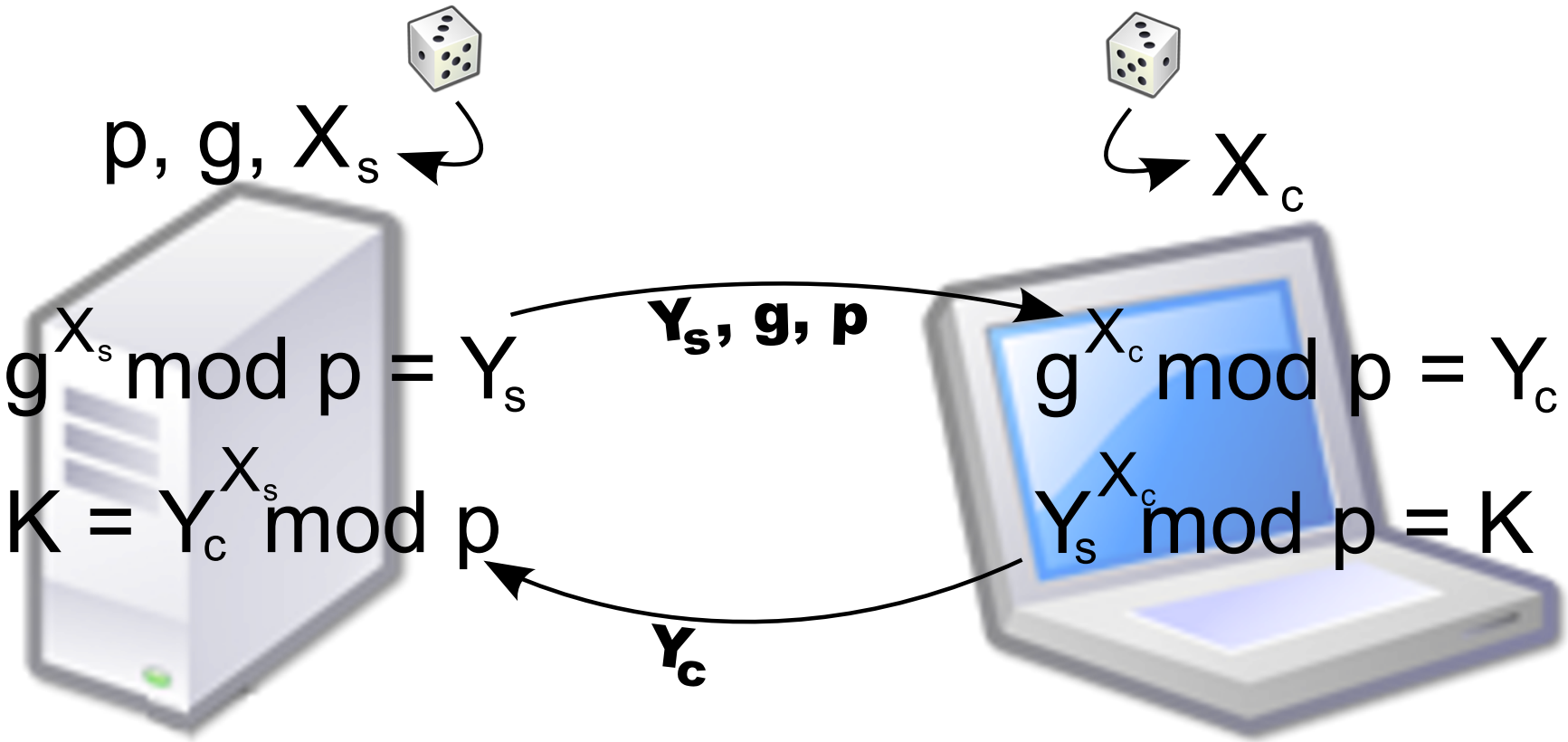
Affected Certificates

May 30th (2 weeks after publication of advisory). Thanks to Juergen Schmidt, Editor-in-Chief Heise Security [www.heisec.de]



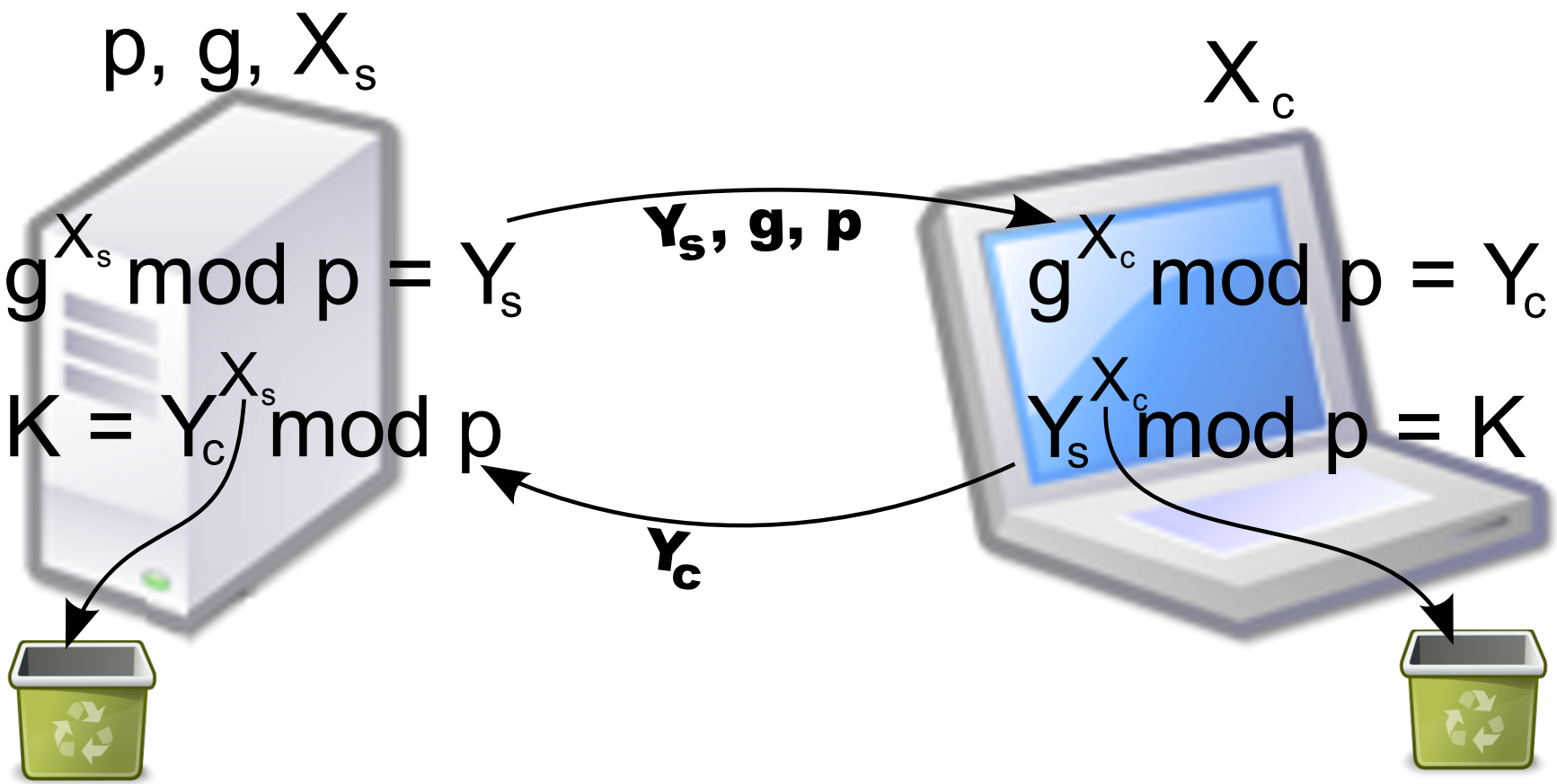
Extrapolating, $\approx$ 24,000 weak certificates signed by a trustworthy CA world wide, over a total of 809,000 sites with valid SSL certificates

Diffie-Hellman Key Exchange



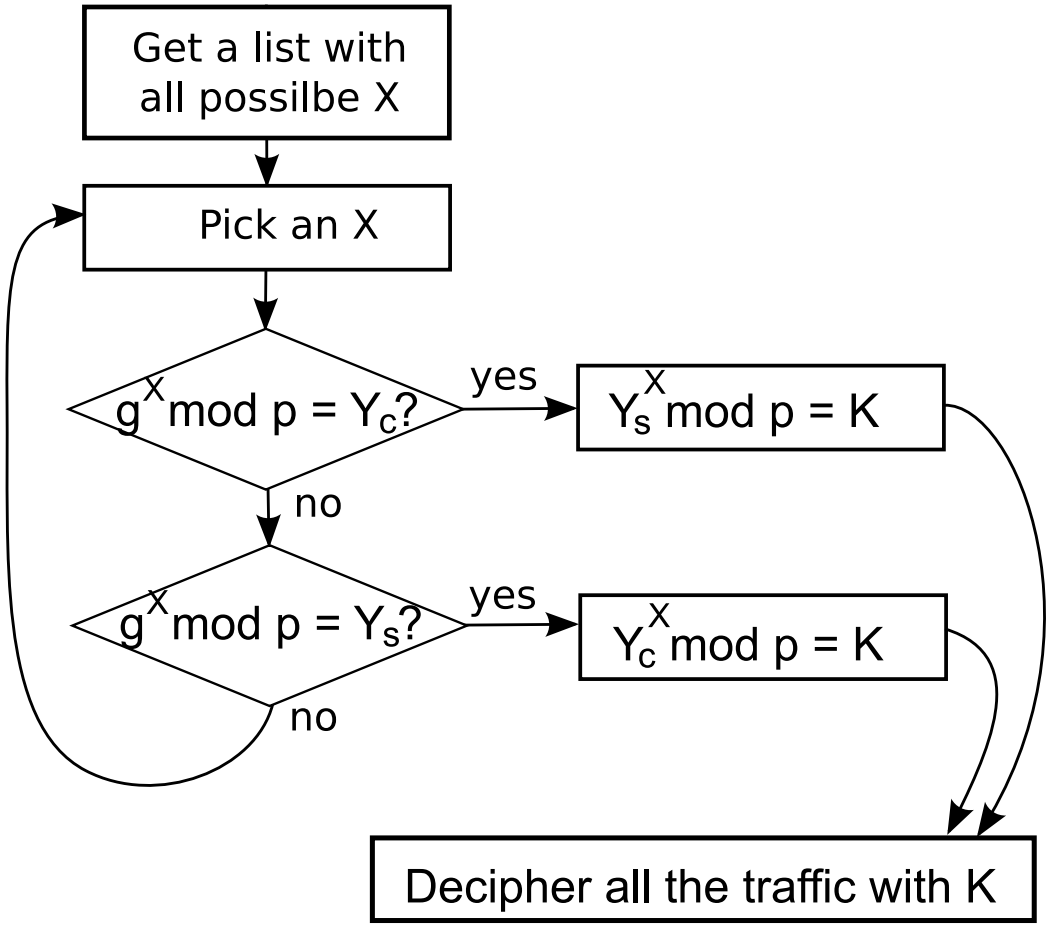
$$Y_s^{X_c} = (g^{X_s})^{X_c} = (g^{X_c})^{X_s} = Y_c^{X_s} = K$$

Diffie-Hellman Key Exchange



Diffie-Hellman Key Exchange

Brute-Force Attack against PFS



public numbers

p, g, Y_c, Y_s

formulas

$$g^{X_c} \bmod p = Y_c$$

$$Y_s^{X_c} \bmod p = K$$

$$g^{X_s} \bmod p = Y_s$$

$$Y_c^{X_s} \bmod p = K$$

Wireshark SSL Dissector/Decipherer

You should be aware of:

- Different apps. can get different private exponents, depending on the PRNG state when calling `RAND_bytes()`
- When not using DHE, there is no PFS, so the attack is trivial
- Most browsers (firefox, konqueror, etc.) implement their own crypto-suite
- Apache forks after the DH key agreement, so the PRNG state changes on each connection

DSA SIGNATURE

The public data is: p, q, g, $y = g^x \mod p$
The private data is: x
To sign a message m:

1. Alice generates a random number, k, less than q.
2. Alice generates:
   - $r = (g^k \mod p) \mod q$
   - $s = (k^{-1}(H(m) + xr)) \mod q$
3. The parameters r and s are Alice's signature for the message m

How Oscar can get x with an explorable space of posible k:

1. Generate r_i with each k_i until $r_i = r$
2. With this k_i: $(sk_i - H(m)r^{-1}) \mod q = x$

Conquer The Universe How-To

1. Choose an application
2. Select or ascertain the algorithm
3. Generate $(2^{15} - 1) \times 3$ keys
4. Brute-force:
   - Authenticate yourself to a server using asymmetric keys (if the server has weak keys for passwordless users)
   - Make a MitM attack using a server certificate clone (you need a weak server certificate and a MitM scenario)
   - Decipher connections made with a vulnerable/outdated Debian peer
   - Attack symmetric-encrypted connections/storage/whatever with randomly generated passwords
   - Get a private key from a DSA user (you need a message signed in a vulnerable Debian)
5. Muahahah!!!

Related Work

- H. D. Moore toys

  (http://metasploit.com/users/hdm/tools/debian-openssl/)
- Wireshark patch to break PFS in SSL/TLS

  (https://bugs.wireshark.org/bugzilla/show_bug.cgi?id=2725)
- SSH Snort plugin by Ben Feinstein

  (http://www.secureworks.com/research/tools)
- SSH tools by Yoann Guillot and Raphaël Rigo

  (http://www.cr0.org/progs/sshfun/)
- Firefox SSL Blacklist Add-on by Marton Anka

  (http://codefromthe70s.org/sslblacklist.asp)
- OpenID/Debian PRNG/DNS Cache poisoning advisory, 08-AUG-2008 (CVE-2008-3280)

  (http://www.securityfocus.com/archive/1/495258)
- Debian wiki. What to do and the effect in each package

  (http://wiki.debian.org/SSLkeys)

COUNTERMEASURES

- Update `libssl`, `openssl` and `openssh` (Haven't you done this yet?)
- Look for and regenerate all compromised keys.
- Eliminate all blacklisted public keys. You can get blacklists from `openvpn-blacklist`, `openssh-blacklist` and `openssl-blacklist` packages.
- Regenerate all DSA keys ever used on affected Debian systems
- Use Firefox SSL-Blacklist add-on
- Use `PermitBlacklistedKeys=no` option in Debian OpenSSH

You can do nothing for the past, the milk is already spilled

Conclusions (aka. lessons learned)

- In cryptography, the algorithms can be strong, but the implementation can be wrong.
- Simple patches may have deep consequences
- Ask with details. “More” is better than “just enough”
- Do not write fancy code, especially if its contribution is minimal
- If many people ask you about something, it may be important
- Get involved in down/upstream patches: you will get improvements
- Linus’s Law counterexample ? (or not... maybe there are *not* so many eyeballs)

Thanks

- Paolo Abeni <paolo.abeni at email.it> for the help with the Wireshark modificaction
- Bodo Möller <bodo at openssl.org> for the help understanding the OpenSSL's PRNG code
- Ignacio "Nacho" Marambio Catán <ignacio.marambio at gmail.com> in the early stages
- Florian Weimer <fw at deneb.enyo.de> for his help in the release of the security advisory
- Kragen Javier Sitaker <kragen at pobox.com> for the code analysis and advice on our (poor) English
- Juergen Schmidt, Editor-in-Chief heise Security (www.heisec.de) for the stats
- 25C3 Organizers, for inviting us. Specialy to fukami.

QUESTIONS?

Thank you!

Luciano Bello <luciano at debian.org>

Maximiliano Bertacchini <mbertacchini at citefa.gov.ar>

POWERED BY LATEX+ BEAMER