148 / juni 2004

DKUUG nyt

Vejen til viden om Åbne Systemer og Internet

Hvorfor er åbne standarder vigtige?

Softwarepatenter - det berører også din virksomhed

cryptar: Secure, untrusting, differencing backup

DKUUG-Nyt er medlemsbladet for DKUUG, foreningen for Åbne Systemer og Internet

Udgiver:
DKUUG
Fruebjergvej 3,
2100 København Ø
Tlf: 39 17 99 44
Fax: 39 20 89 48
email:
dkuugnyt@dkuug.dk
Sekretariatet er åbent:
Mandag - fredag
kl. 10.00 - 15.00

Redaktion:
Hanne Vilmann
(ansvarshavende)
Peter Toft
Sidsel Jensen
Keld Simonsen
Henrik L. Kramshøj
Kristen Nielsen

Tryk:
Palino Print

Annoncer:
Kontakt DKUUGs sekretariat

Oplag:
1500 eksemplarer

Deadline:
Deadline for næste nummer nr. 149 er den 28. august 2004

Medlem af Dansk Fagpresse

DKUUG-Nyt
ISSN 1395-1440

Indhold:

Leder:

Det er blevet svært at være bladudgiver.
pr. 1. marts 2004 er portostøtten fjernet fra blade, dette betyder med postdanmarks forhøjede priser at det bliver tre gange så dyrt at sende DKUUG-Nyt ud.
Vi forsøger at få vores blad med i en puljeordning som gør det lidt billigere, men sagen afventer en EU afgørelse.

Vi bringer i dette nummer og næste nummer af DKUUG-Nyt 2 udvalgte papers fra NordU konferencen.
Selve konferencen blev desværre aflyst grundet for få tilmeldinger og grundet aflysningen kunne de 2 papers ikke publiceres på selve konferencen. Men vi mente det alligevel kunne være interessant for DKUUGs medlemmer og DKUUG-Nyts læsere at stifte bekendskab med.

Som noget nyt har vi indført formandens side.
Her kan i få et indblik i hvilke visioner og tanker som driver vores formand.

DKUUG er begyndt at udsende nyhedsbrev på mail, vi bringer i dette nummer de to nyhedsbreve som har været udsendt.

Vi ønsker alle en god læsning og en rigtig god sommer.

Hanne Vilmann

A system for managing sensitive machine configuration information on a large-scale.

Authors

Vladimir Bahyl, Thorsten Kleinwort, Lionel Cons, Jan van Eldik, Tim Smith

Abstract

This paper discusses the design and implementation of a system for a largescale management of sensitive machine configuration information. By sensitive machine configuration information we mean password and configuration files as well as encrypted key-pairs.

The paper doesn't talk about yet another account management system nor it describes ready to be installed tool. It rather inspects our unique approach in building the trusted relationship with bare nodes in a large cluster and distributing sensitive information to them.

We explain the problem in depth, we mention constraints we had to face, we describe tools we used for our solution and we also dedicate a significant part of this paper to implementation details. As a part of the conclusion we mention possible enhancements for the future.

Introduction to the problem

There is a security hierarchy of the information in every operating system. It means that there are certain data files that have to be kept safe, not accessible by ordinary users, in order to guarantee an expected behaviour of the system.
These files can contain (usually encrypted) passwords, configuration settings (e.g. /etc/sudoers file or Windows registry data), access control lists, host specific information (e.g. SSH host key pairs) and others.

How this information gets transferred onto the node usually depends on the whole process how the node is installed.

On a small scale (tens of units), when nodes are installed on one-by-one (or byhand) basis, there isn't usually a problem of transferring this important data onto them. It can be read from a safe media (floppy inserted by a trusted operator, personal smart card, etc.), transferred or even typed in, via a secured channel (locally attached keyboard). Every time, there is a need for modification, an operator can go to each node and correct the files. The security in this case is based on the fact that the operator is trustworthy and loyal to the organisation.
However, the situation becomes lot more complex, once the scale of the computer farm reaches hundreds of nodes. These, initially bare machines, without any pre-installed operating system need to get to the same state as if installed by a person. However, giving the high number of them, the installation should be fully automatic. On top of this, in an organisation with a large network infrastructure and with an open policy towards internal users, unregistered devices that violates internal network security has to be expected. Hence we get to the point where we have to split the problem into 2 distinctive parts:

- Initial trust building between the central repository and a node
- Distribution of modifications across the farm

Initial trust building between the central repository and a node:

Core of the centralized security model (used as a background in this paper) is a well-protected place that contains all sensitive information. In computer terms, this is usually a server with access highly restricted – let's call it central repository. As the farm nodes are installed, they contact (or are contacted by) the repository to obtain the sensitive data. This access to secure

information must be obviously based on a trusted relation.

However, as mentioned above, the nodes are initially not running any operating system, thus not holding any unique identification. They have to be provided with one – preferably during the installation.

Nowadays, following 2 types of the installations are used for setting up of many farm nodes:

- Image cloning
  With this approach, when a disk image is prepared on a central node, the unique identification can be directly part of it. It is then securely transferred to the node and as the node loads it, it can start using it.
- Complete installs
  Here, the node first gets installed and then asks for its identifier. This is the approach this paper will concentrate on, as this is the way we install machines.

In any case, it would be very difficult to build the trust out of nothing. That is why, in order to reduce the problem space, we were forced to assume certain trusted and independent data sources within our internal organizational infrastructure.
Most important are:

- DNS records
- Centrally managed network and configuration databases (use Oracle)

Both those places have access highly restricted to only small trusted group of administrators. It means that we consider the information returned to us by our internal DNS as well as network and configuration databases to be correct.

Distribution of modifications across the farm:

Once the nodes are installed and trusted by the central repository, it is relatively easy to provide them with the required secure data. But still, the transfer must be:

- Fast
- Secure
- Reach all nodes

In fact, the importance of the modification distribution task rises directly with the importance of the file that needs to be modified. For example, if the root's password needs to be changed everywhere for example after a security incident, the system administrator has to be able to do this quickly and ensure that all selected nodes are reached. Only fast and accurate action will help him/her to minimize exposure to the system when they are most vulnerable.

On top of these requirements we also choose that each transfer use a unique channel for more security.

Comparison with prior work

Browsing through the prior art exposes plenty of work in various relevant areas.

Already a brief search on the Internet (or in the papers presented at previous USENIX conferences) reveals plenty of work dealing with effective account management. There are many systems out there of various qualities and purpose for password file handling.

Similarly, [2] talks interestingly about difficulties of establishing the genuinity of a remote computer.

However, according to our knowledge, none of these works focus on a problem of building a trust with a bare node and creating a secure data channel.

Tools we decided to use

As many people out there we choose to use open-source tools to meet our goal.
It turned out that:

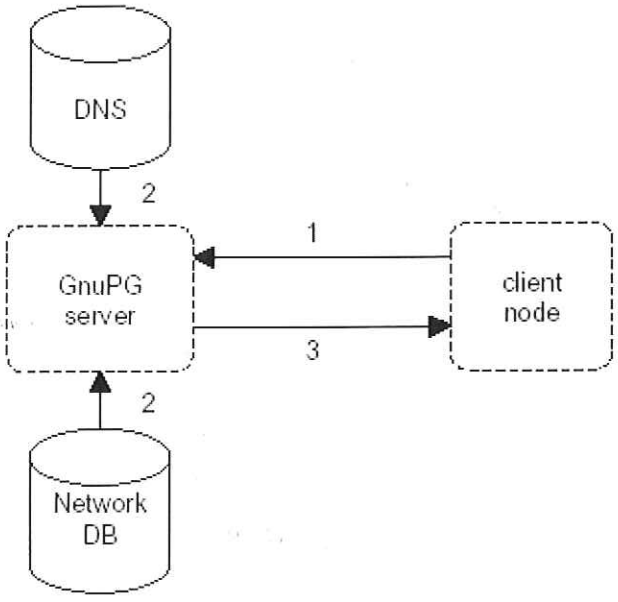
- GnuPG would suit our needs for a general file encryption
- HTTP is fine, low overhead protocol, for transfer of the encrypted files
- SSH (most notably scp) is perfect for some additional transfers requiring additional care

How do we build the trust

Simply said – we use GnuPG key-pair as the identifier. Prior to the installation, this key-pair, unique to each node, is created. It is generated and stored in the central repository – a GnuPG key-pair server with access highly restricted.

Then, as the node is being installed, the installation script will connect to the server and download the key pair (using Secure Copy - scp).

Schema of the whole trust building process is on the following figure:



1. Installation of a client; it will open a scp connection – ask for it's own GnuPG key pair.
2. GnuPG server will compare requestor's hostname and reverse looked up IP with records in the DNS and the network database.
3. GnuPG will serve key pair over the scp connection.

A careful reader might notice some simplification here, but as previously stated, in order to be able to cope reasonably with the problem, we trust the internal DNS records.

This is an example of the scp command with all actually used options:

```
scp   -oProtocol=1
      -oStrictHostKeyChecking=no
      -oUsePrivilegedPort=yes
      -oRhostsAuthentication=yes
      user@server.cern.ch:path/client.cern.ch-key-pair.tgz
      /tmp/tmp-dir-11001
```

The connection above from the client to the server is possible only during a certain time window. By default, this window is opened for 3 days and can be reactivated by selected group of people only.

To explain the options to the scp, one has to remember that the installation itself is fully automatic. That is why:

- We have to set StrictHostkeyChecking=no as there is no-one to check and say `yes´ (~root/.ssh/known-hosts file is empty anyway as it is being installed)
- We have to use RhostsAuthentication because of the time window setup (list of hosts allowed to connect is stored in .shosts file of an unprivileged user)
- We have to have Usepriviledgedport=yes as a complement to the type of the authentication described above
- For various reasons we still use SSH protocol version 1 on our site

After the transfer, the key-pair stays on both: client and the server. Server uses its public part to encrypt the sensitive data for a specific client. Client is then using its private part to decrypt the message. Obviously, server keeps all individual key-pairs of all nodes for this reason.

In case of a suspicion that the GnuPG key-pair was exposed to an unworthy individual or the node changed its purpose a new key-pair is re-generated and the node is re-installed from scratch.

One might argue here that we could make the system stronger by generating one-time GnuPG key-pairs only. Like this, there would be no need for a time window, as it would be possible to download the key-pair only once. Although this could be done in the future, we choose to re-use the key-pairs mainly because of the frequency with which we re-install nodes while still having to do some manual steps in the installation preparation process (key-pair generation being one of them).

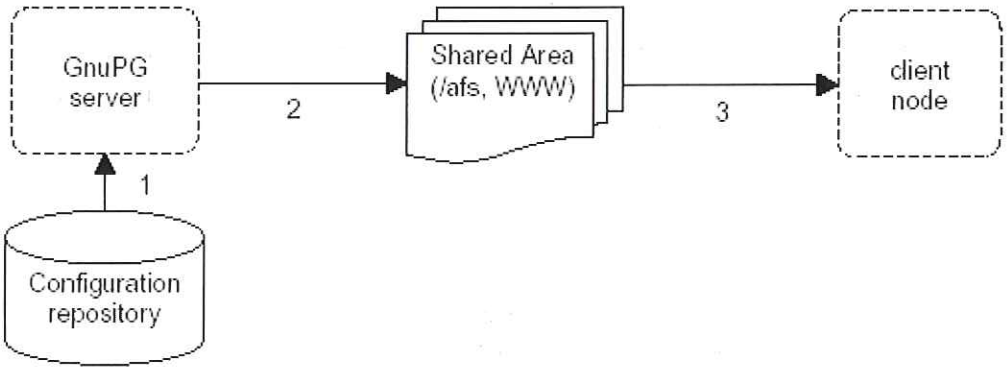
Another interesting point to a discussion here would be: why don't we distribute a server key to the clients as we could then use that key to authenticate (if it is SSH identity key) or decrypt messages from the server (if it would be GnuPG public key). Although, this would be feasible to do in the beginning of the trust building process, we didn't see any possibility of gaining more security. Later, when each node has it's own GnuPG key-pair this would not be necessary.

However, incorporating servers SSH identity key into the initial connection process might be necessary when we move from protocol version 1 to version 2.

Once the key-pair is on the node (as well as in the central repository on the server), the trust has been built.

Key-pair as the dedicated communication channel

The process of the transport of some sensitive data from the GnuPG key-pair server to the client is described on the following figure:



1. GnuPG server will pull sensitive configuration data out of the configuration repository.
2. GnuPG server will encrypt the data with public key of the given node and publish the encrypted file in a common area.
3. Client will download the prepared data file, decrypt it with the secure key and use it locally.

As you can see, there is no need for additional security; the transport channel can be implemented via simple HTTP protocol, AFS or NFS areas, because the information is already encrypted. This saves usual secure tunnelling overhead.

How do we distribute the modifications

So far, we have dealt only with the initial trust building and first transport of the sensitive data. Another question is of course, how to distribute the modifications that have been done in the configuration repository.

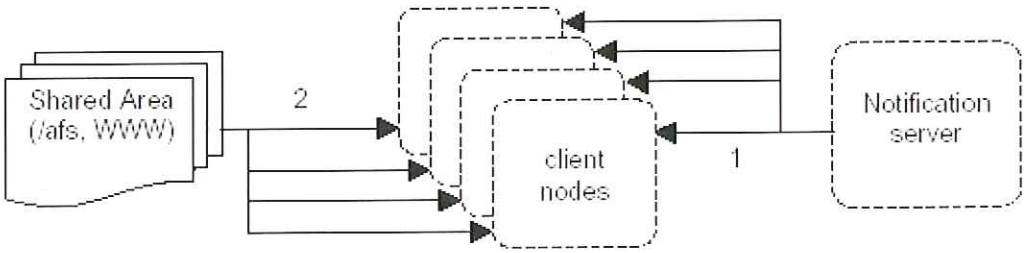
As far as updates are concerned, they are delivered to the nodes via a simple *triggered-pull principle*. When triggered with a specific tag, client executes a selected script and pulls the data from the configuration repository. For this – additional notification mechanism (described further bellow) is running and listening on each node for a small set of tags.

This mechanism is widely used in our farms (not only for sensitive data transportation). Its main advantage over the true push mechanism is that there is no overhead with keeping a list

of all nodes where to push on the server. All clients are configured with just a server name to which they subscribe for the notification.

Prior to the notification, modified files for all selected nodes have to be prepared on the servers. Hence on the GnuPG server, the changed data are encrypted with public keys of all affected nodes. Once this is done, the nodes can be notified.

On notification (after the GnuPG server has modified and prepared sensitive data), each client will go to the central repository and transport again its uniquely encrypted copy of data. Then locally, each node will process it with an appropriate script (for an example, see further bellow).



1. After the modified configuration files have been re-encrypted with the individual public keys of the nodes, the notification server can notify clients about the change.
2. Each client will then fetch its copy from the central repository and process it with appropri ate script locally.

To prevent overload on the repository, clients wait for a short random period of time.

Implementation details

To describe more closely our system, I would like to mention some details of the key scripts as well as options that we use so our solution can be easily re-applied somewhere else.

As for implementation tools, all components of the system have either been written in or put together with Perl. The server part is running on nodes under Linux. The majority of the clients are also Linux boxes (more than 1000) but we also have around 20 Solaris ones.

GnuPG key-pairs generation

Currently we use *gpg (GnuPG) 1.0.7* package to generate the key-pairs. To make the generation process as automatic as possible and because gpg binary is normally only used interactively, we have to use an experimental feature which allows to create keys in batch mode.

1. Fist we create a configuration file ($CFGFILE) with options for the generation:

```
%echo generating for $HOSTNAME (IP: $IP)
Key-Type: DSA
Key-Length: 1024
Subkey-Type: ELG-E
Subkey-Length: 2048
Name-Email: root@$HOSTNAME#$IP
Expire-Date: 0
```

```
%pubring $DSTDIR/$HOSTNAME.pub
%secring $DSTDIR/$HOSTNAME.sec
%commit
%echo done
```

2. Then we call gpg binary ($GPG=/usr/bin/gpg) like this:

```
$GPG --batch --gen-key $CFGFILE
```

This creates 2 files:

```
-rw------- 1 root root 1157 Feb 25 13:59 HOSTNAME.pub
-rw------- 1 root root 1726 Feb 25 13:59 HOSTNAME.sec
```

that are then later delivered to the node over the previously described scp connection during the installation process. Files are put into ~root/.gnupg directory as:

```
-rw------- 1 root root 1157 Feb 25 17:05 pubring.gpg
-rw------- 1 root root 1726 Feb 25 17:05 secring.gpg
```

When this is done, node can use this key-ring to encrypt and decrypt data.

Use of the encrypted data locally on the node

In this paper we have already mentioned several times the fact, that the encrypted data on the server are transferred and used locally on a node. Now, I would like to give you some examples about how this is done.

Currently, we use this mechanism to transfer following sensitive files:

- Header part of the /etc/passwd and /etc/group files that contain encrypted password of all local accounts on the node (AFS accounts of the users are handled by a separate mechanism which is outside of the scope of this paper).
- /etc/ssh_host*key* files – especially needed for our interactive clusters where all nodes in the cluster must have same files in order to look the same for SSH. (As nodes are re-installed, this allows us to bypass the man-in-the-middle warning from SSH when public key of the node has changed.)

Please let me describe the 2nd example, as it is more interesting:

1. First, encrypted TAR file of all /etc/ssh_host_*key* files is transferred over the HTTP from the repository locally using LWP::Simple Perl library.
2. This file is then decrypted with this Perl call:

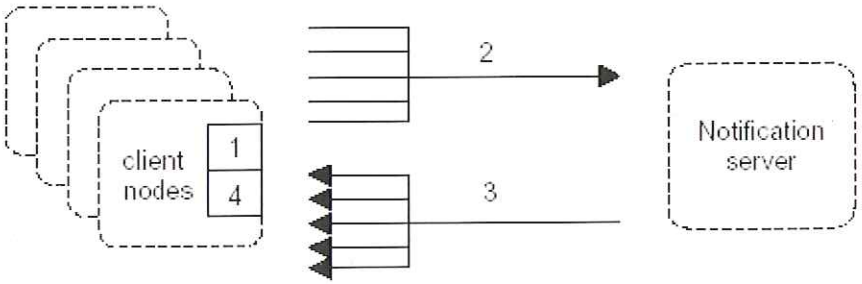
```
system("su - root -c 'umask 077 ; /usr/bin/gpg -q
--yes --batch --local-user root\@$hostname --output
$tmpfile --decrypt $infile'")
```

3. It is necessary to use the su command in order to get full environment of root.
4. Decrypted file is extracted with tar xfz command and individual files are put into /etc/ssh directory with appropriate names.

A word about the notification

Even though, the notification system itself is quite complex, I would like to describe it a bit here, as it is a very important component. Its main advantage is that it allows rapid modification deployment.

Its functionality has already been mentioned earlier and it is described by the following schema:



1. A notification daemon QRWG (written in C) is started on each node.
2. It looks into its configuration directory and subscribes for each tag to the notification server.
   Here is an example of such a configuration file for /etc/ssh_host_*key* files deployment:

```
ssh_host_key {
runatboot=yes
subatboot=yes
subatcron=yes
server=notification.cern.ch
exec=/usr/local/sbin/get_secure_file
priority=100
}
```

3. The daemon on each client then waits for a notification (in the example above it must arrive from notification.cern.ch and the tag must be ssh_host_key) from the notification server.
4. Once an expected tag is received, argument in exec parameter is called (/usr/local/sbin/get_secure_file in our example). It is normally a script that will execute the appropriate action.

Tags are transferred via a short TCP connection or carried by an UDP packet.

Notification is send to clients on demand (when we know there is a change that needs to be distributed) but also (if configured) nodes execute the action scripts on boot. This synchronizes them with others if they were unavailable for whatever reason.

Real experience and future plans

Prior to deployment of this new system in our organisation, we used to have scattered solutions to the different problems mentioned above. Various custom scripts did modifications to header part of the password file. Specific scripts were developed for each case. Frequently – machines were unavailable – making synchronisation labour intensive once they were back. Distribution of /ect/ssh_host_*key* files was done with (yet another) script where password for decryption was read from an obscure place.

Nowadays, described universal data management mechanism allows us very quickly distribute changes of sensitive data. As a large organization with outsourced management of the computer centre there is a significant fluctuation of operators. Not to mention occasional security incidents.

With help of this system, we are able to change for example root password on the whole farm

of more than 1100 nodes within 15 minutes (including configuration database modifications).

Even though this sensitive data management system suits sufficiently our current needs it could be improved. Some of the possible improvements would be:

- Make DNS queries on the GnuPG server over encrypted connection (SSL)
- Distribute an identity file with initial GnuPG key pair installation script RPM so that clients can identify themselves. However, the time window mechanism would still be necessary, as anyone who stole the identity file could get the GnuPG key pair.

Conclusion

This paper tries to address a general problem – how to distribute sensitive data to a large number of clients in a secure way. Currently it is mainly used for distribution of the local accounts and passwords, but the solution is universal and it would now require very little effort to extend it to distribute another sensitive data to clients if necessary.
We believe it might already be useful in an environment with just tens of similar clients.

References

1 Free Software Foundation. GnuPG. http://www.gnupg.org, 2003.
2 Rick Kennell and Leah H. Jamieson. Establishing the Genuinity of Remote Computer Systems. In *Proceedings of the 12th USENIX Security Symposium*, pages 295-310. USENIX Association, 2003.

Tina Bird var et af de helt store trækplastre under NordU 2004

Foto: Kristen Nielsen

cryptar: Secure, Untrusting, Differencing Backup

Jeff Abrahamson
Department of Computer Science
Drexel University, Object Recognition and Applied Algorithms Lab
Philadelphia, PA
jeffa@cs.drexel.com; http://www.cs.drexel.edu/òjeffa/
Adam J. O'Donnell *Department of Electrical and Computer Engineering*
Drexel University, Computer Communications Laboratory
Philadelphia, PA
adam@ece.drexel.com; http://www.ece.drexel.edu/CCL/odonnell/

Abstract

We present an algorithm and its implementation to perform secure incremental backups to untrusted servers.
The algorithm avoids unencrypted data leaving the local host, but is nonetheless able to compute a set of differences between the encrypted data on the remote server and the changed local files.

The `cryptar` algorithm is derived from the rsync algorithm.
It is conceptually similar to running rsync with the remote image encrypted.

1 Introduction

Traditional backup techniques focus on writing a set of files (or a subset in the case of incremental backups) to tape or other removable media that may then be kept in a safe place. Incremental backups typically write the complete content of each changed file to removable media.

Inexpensive local storage and storage-intensive non-text content (audio, video, images) have made this practice more expensive: a typical user has much more data than a few years ago. A small business or individual may easily have such a large quantity of data on a $100 hard drive that tape storage becomes more expensive than the original hard drive.
Moreover, proper off-line storage practices require a second, remote site, something few people do in practice.

For these reasons we have developed a new remote backup tool called `cryptar`, which allows a remote computer and its hard drive to be used for secure full or incremental backup. The key innovation in `cryptar`, however, lies in keeping all data in the remote store encrypted, while still being able to compute efficient deltas so as to conserve network resources.

We do this by maintaining a very small amount of local meta-data, about one hundred bytes per file, while storing nearly all other information remotely. We will describe the precise details in section 3.2 ff.

This work was supported in part by the United States National Science Foundation Graduate Research Fellowship.

1.1 Why `cryptar` is Interesting

The principal historical problems related to remote online storage have been bandwidth and security. Most people who consider backup at all, though, now have low-latency broadband connections that make maintaining an off-site backup feasible.
This leaves us, however, with concern for the safety of data stored remotely. Unless one completely trusts the remote machine, a serious concern remains that the remotely stored data may be perused by a curious (or evil) person with access to the remote machine.

`cryptar` solves the data integrity and privacy problems by encrypting the remote content before transmission while storing a small signature locally. Indeed, `cryptar` stores each local file remotely in encrypted blocks, then keeps a list of hash signatures of those blocks to permit comparing blocks without needing direct access to them.

`cryptar` is thus able to compute the necessary updates without extensive access to the remote and encrypted data. This permits efficient low-bandwidth updates without allowing unencrypted remote data.

1.2 Trust

If we trust the remote site completely, we can use rsync snapshotting [Rubel] or CVS. We would like to imagine a world, however, in which we are not obliged to find secure machines for our offsite storage needs. Already we can purchase inexpensive disk space from many ISP's. Except for the questions of trust and efficiency of data transfer, we could use this space for our backups.

The principal problem with maintaining backup data on insecure servers, however, is the risk that the information is read by unauthorized parties or even altered without our knowledge. We can solve the first of these problems by encrypting the data, and the second by signing the data we store. But encrypted backups typically do not admit efficient delta computations, since similar files are generally dissimilar once encrypted.

The problem we solve with `cryptar` assumes that secure and authenticated data transport is easy (e.g., IPsec or ssh), and that computational power is cheaper than bandwidth. (If bandwidth is cheaper, we can simply transfer full encrypted files without concern for efficient deltas.)

In `cryptar` we encrypt blocks before transfer to the remote archive while keeping block signatures locally.

Because the remotely stored blocks are encrypted locally before transfer, only encrypted data leaves the local host.
A remote intruder would find in the archive only encrypted data, useless up to the strength of the encryption scheme.

In addition, the local store maintains a set of SHA-1 hashes for the remote data. Any data retrieved is therefore compared against the local hashes to authenticate it.
A more detailed discussion of the strength of this assurance resides in section 3.7.

1.3 How cryptar is Different

`cryptar`'s principal innovation is the combination of keeping remote data encrypted and yet being able to compute differences efficiently.

Rsync, for example, which is discussed in section 3.1, computes differences efficiently, but must have access to unencrypted remote data. A number of programs encrypt
remote data but are incapable of computing an ef-ficient delta.
Indeed, the poor man's solution to encrypted remote backup has been to encrypt data to tape or to a remote machine by simply piping the output of the unix tar program through an encryption program (cf. section 2).

We will discuss related work to further understand how `cryptar` is different from other cryptographic and archive software and algorithms. After a brief discussion of trust and who we do and don't trust, we discuss the actual algorithm in depth. Finally, we'll discuss security, the `cryptar` protocol, and make some short notes on future directions.

2 Related Work

A venerable solution under unix-like operating systems is tar | gpg | dd (with various options to make this make sense)[1]. This effectively encrypts the remote data before it leaves the local machine, but provides no ability to compute deltas or to update the remote store without recomputing the entire remote image. Using find or even tar, incremental backups are possible, but only in the sense that an entire file that has changed since the last backup can be re-encoded. We are concerned here that the files we recover are what we originally stored: this scheme provides no such guarantee.

It's interesting to note that the following ad hoc remote archive scheme fails: For each file foo, encrypt foo to foo.crypt, rsync foo.crypt to the remote server, then delete foo.crypt locally. Remote files are always encrypted, and one could easily add signing to this scheme. Unfortunately for this scheme, encryption preserves exceedingly little structure, and so rsync cannot compute a delta smaller than the file itself. Rsync is thus forced to copy the entire file, which makes this scheme too inefficient for our needs.

The rsync program allows for incremental backup, albeit via a somewhat cumbersome interface.
RIBS, the rsync incremental backup system (http://rustyparts.com/scripts.php),
provides a nicer interface around incremental backup with rsync, but the remote files remain unencrypted. No certificate guarantees that a recovered file is correct. The
rsync-backup program (http://www.stearns. org/rsync-backup/) and Scylla Charybdis (http://www.scylla-charybdis.com/) suffer the same limitations.

Duplicity (http://www.nongnu.org/duplicity/) stores GnuPG-encrypted tar archives on one or more remote machines. It uses GnuPG to sign the archives in order to know that what it gets back is what it sent. On the other hand, it maintains a great deal of data in its (local) incremental storage space, and the size of the backup can far exceed the size of the original files if the files change often.

---

1 gpg is GNU Privacy Guard, an OpenPGP compliant crypto program. The UNIX program dd copies files.

The Distributed Internet Backup System, DIBS (http://www.csua.berkeley.edu/òemin/source_code/dibs/) maintains GnuPG signed and encrypted archives on one or more remote servers.
It uses Reed-Solomon codes to distribute data over multiple servers. When a file changes, however, the entire file must be retransmitted.

A number of P2P systems offer superficially similar services. Chord/CFS (http://www.pdos.lcs.mit.edu/chord/) and Mnet (http://mnet.sourceforge.net/), for example, both provide encrypted network storage. These P2P systems, however, do not compute efficient differences.

Plan 9's Venti block store [Quinlan] uses a SHA-1 indexed block-based differencing scheme to store files. The goal of Venti (enterprise file storage) is quite different from rsync or `cryptar`, but the block-based differencing scheme is similar.

The Self-Certifying File System (http://www.fs.net/), SFS, operates much like NFS, but encrypts its traffic. The user need not trust the network, but must trust the remote host with his unencrypted data, since the encryption only covers transport, not storage.

3 Algorithm

It is useful to begin with a brief explanation of the rsync algorithm. We will then describe the `cryptar` algorithm in overview, define some terms useful for a more detailed analysis of `cryptar`, then describe the specific steps involved in archive and restore operations. We will conclude our discussion of the algorithm with discussions of the signature algorithms, our confidence in those signatures, and the cryptographic framework for remote block protection.

3.1 The Rsync Algorithm (Overview)

Rsync is a client-server binary differencing algorithm designed for contexts where bandwidth is more expensive than computing time. The goal is to transform a remote file such that it is identical to a local file. It uses checksums to determine where changes have occurred.

Rsync depends on an easily computed 32-bit checksum, called the weak checksum, and a stronger but more expensive cryptographic hash called the strong checksum.

Given the two files, the client requests that the server compute both the weak and the strong checksums of each B-byte non-overlapping block of the remote file R. That is, it computes the checksums of the byte range $(0,B-1)$ of $(B,2B-1)$, and so forth. The window size B is a constant of the algorithm, although Tridgell [Tridgell2] discusses theoretical and practical optimal ranges for this constant.

The server sends this signature (the set of weak and strong checksums) to the client. The client computes *every* weak checksum of the local file. That is, it computes weak checksums for intervals$(0,B-1)$, $(1B)$,$(2,B+1)$, $(3,B+2)$, etc., see figure 1. The client then uses a hash of these weak checksums to find candidate matches to the server side checksums.

The client computes a block's strong checksum only to confirm a weak checksum hit. A weak checksum miss is clear proof that the blocks are not identical.

Knowing which blocks are identical between the local and the remote files, the client can instruct the server to move blocks as needed, transmitting to the server only that data that is

not already represented in the remote file according to this block view. The process is symmetric for file recovery.

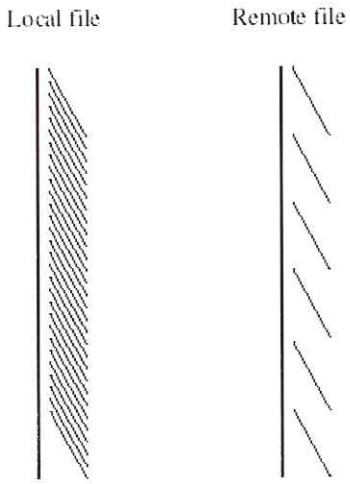


Figure 1: In rsync the remote file (here the thick line on the right) is covered by disjoint blocks (diagonal lines) while the local file (the thick line on the left) is covered by blocks at all offsets. In `cryptar` the same holds, but the remote file's covering was stored in the database at the time of archive, and so is recovered from the database rather than computed from an actual remote file.

3.2 The `cryptar` Algorithm (Overview)

cryptar uses a split database to simulate a remote server with access to remote files. The remote `cryptar` process acts merely as a block server.

As in rsync, a local `cryptar` process establishes communication with a remote cryptar process whose only difference is the method of invocation: the remote process runs in server mode. The details of communicating are at the user's discretion, but might typically involve an ssh connection.

The essential idea behind the cryptar algorithm is to store encrypted data remotely, but only a small amount of meta-data locally needed for interacting with the remote.
We thus store in the local database a signature as well as a small amount of bookkeeping information.

Of course, while the purpose of rsync is compressed copy, the purpose of cryptar is secure archive. The remote file is actually virtual: it exists as an image in a split database, the remote database being encrypted.

Figure 2: A full minimal covering

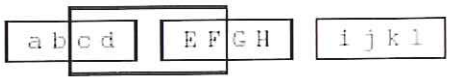


Figure 3: A full non-minimal covering

Said another way, the local database stores enough information to simulate the response of an rsync server if we were running rsync. This local meta-data is what allows the remote data to be kept encrypted.

The client may thus retrieve the virtual remote file's signature from the database, compute the

delta, and store the delta in the database. By storing almost all the information encrypted remotely, the local database can remain small enough to avoid burdening the local system's storage, but still hold enough information to permit us to avoid querying the remote database unless a file has in fact changed.

3.3 Definitions

We call the local set of files which we want to archive the *fileset*. The data that we store remotely we call the *remote (archive) database*.

A *covering* of a file is a set of possibly overlapping pieces of the file that, taken together, completely specify the file's contents. A *partial covering* is a set of possibly overlapping pieces of the file that fail to specify the entire file (i.e., that leave holes in the file). We will only concern ourselves with coverings using fixed size blocks for efficiency reasons.

To illustrate coverings, consider a 12 byte file (string) abcdEFGHijkl. An example of a minimal covering (Figure 2) would be bytes 0-3 ("abcd"), bytes 4-7 ("EGFH"), and bytes 8-11 ("ijkl"). Figure 3 shows a non-minimal covering of the same file.

If we insert two bytes after byte 2 ("abcxydEFGHijkl"), the last two blocks above only form a partial covering (figure 4): "EFGH" and "ijkl" cover part of the file, but "abcxyd" is uncovered. We can add two more blocks, "abcx" and "ydEF" to form a full covering, as in figure 5.

Figure 4: A partial covering

Figure 5: Two partial coverings whose union is a full covering

In section 3.5 we will describe a scheme for computing and storing weak checksums and strong checksums for each file. We call the set of weak and strong checksums of the blocks in the covering the *block list*. The statistical summary of the file (its length, a SHA-1 hash of the file, and its modification date) we call the *summary signature*. The summary signature and the block list together we will sometimes call the *signature* of that file at a given time. The *local database* stores the summary signature; the block list we store (compressed and encrypted) remotely. Table 2 shows an example of a signature.

Local DB	Remote DB
Filenames	File blocks
Summary signataure	Block list
(Keys)	

Table 1: Summary of what is stored where. Note that the remote database has no knowledge of the structure of its blocks, it merely stores key-block pairs. As noted in section 3.6, the key table is currently omitted, but the implementation foresees it´s possible use.

3.4 Walk-through

3.4.1 Archive

Given a file in the fileset, either it is already represented in the local database or it is not.

If it is not represented, this is the first time we have seen this file. We compute the file's signature, store the summary signature locally, and store the file itself, in compressed and encrypted blocks, and the block list remotely.

If the file is represented in the local database, we can compute, by comparing the stored and actual modification date and SHA-1 hash, whether it has been modified since the last archive pass. If it has not been modified, we are done. If it has, we compare it to its last stored signature, compute a partial covering using blocks in the archive (computed based on the signature), and finally compute a second partial covering that fills the holes left by the first covering. We update the summary signature in the local database and send the new partial covering and the new block list, both compressed and encrypted, to the remote database.

Summary Signature:

```
Length:  351824 bytes
File SHA-1:  6854 7369 6920 2073 6F6E
             2074 2061 4853 2D41 2E31
Block list SHA-1:
         63C6 16BF 8C4C 0D1D AE4B
         1A1A BA95 D082 A558 A276
Modification date:  Sun Nov 17
                    21:46:29 EST 2002
```

Block List:

block id	offset	weak sum	SHA-1
23867	0	759E AF5E	6979 2969 C569 92FE 6572 FE17 2D9F EFC6 FBFF BCBF
23868	2048	1B76 1BB8	27C1 0BC1 A7C1 EF22 7F0D 3B04 7B82 BF82 E882 82FC
23869	4096	B565 07BC	63BB 9617 7D89 F290 F761 BE96 3D8F 57DA 13EC FB8F
⋮			
about *NIL* blocks: in this example L=2048			

Table 2: Example signature of a file. In practice, some additional bookkeeping information is kept in the signature as well.

3.4.2 Retrieval

To recover a file, we look up its summary signature in the local database and fetch the block list from the remote.
We then determine the block identifiers(2) to request from the remote store, request them, and then reassemble the file locally.

(2) The block identifiers are assigned sequentially by cryptar and serve as primary keys for blocks.

Reassembly, of course, means that we decrypt and uncompress the blocks we have received. During reassembly we check the hashes from the signature against the retrieved blocks to verify that the blocks we have recovered are, in fact, those we sent.

If we are recovering a file that already exists in some version locally, `cryptar` may use the local copy for recovery of those blocks unchanged since the last archive operation, and so avoid transferring the entire file.

It's worth noting that we can determine the authenticity of the retrieved blocks by means of the SHA-1 hash that we have stored locally for each block. In order for an attacker to give us forged data, they would have to know not only the SHA-1 hash that a block must match, but also give us blocks of known length that match our given SHA-1 hash.

The reconstructed file must also match a pre-computed SHA-1 hash. But SHA-1 [FIPS-180-1] is a 160 bit cryptographic hash designed specifically to make this hard.

3.5 Signatures

As in rsync, we compute an inexpensive but weak rolling checksum of every block in our file (cf. figure 1). That is, given an N-byte file *F = (b1, b2,...., bn)* , we compute a total of *N - L* weak checksums, where is the block size: csum *(b1,...., bl)*, csum *(b2,...., bl+1)*, and so forth.

We create a hash table of weak checksums of every such -byte long block of the file. At this point rsync would ask its server to compute checksums of a minimal covering of the remote file. The block list that we stored at the time of the previous backup contains the checksums for a covering of the file. This allows us to compute the delta without access to the remote file.

Note that the set of weak checksums computed on the covering is a subset of the full set of weak checksums that we compute above.

The weak checksum we use is the rsync weak checksum based on Adler32 [Deutsch] (but in this form due to Andrew Tridgell, who in turn credits Paul Mackerras [Tridgell2]):

$$r_1(k) = \left(\sum_{i=0}^{L-1} a_{i+k}\right) \bmod M$$

$$r_2(k) = \left(\sum_{i=0}^{L-1} (L-i)a_{i+k}\right) \bmod M$$

$$r(k) = r_1(k) + Mr_2(k)$$

Here *k ¤ {0,...,N -L}* is the offset in the N-byte long file of the beginning of the window on which we are computing a checksum, and is the size (length) of the window.

We use *M = 2/16* for efficiency of computation. Note in particular that the sums overlap, and so computing a block's checksum *r(k)* once we know the checksum for the block offset one byte earlier, *r(k-1)*, requires only a few arithmetic operations.

For each block in the remote file, whose checksum we know from the block list, we look up the weak checksum in the hash table to determine where it might fit in a partial covering of the local file. If we don't find a hit, we may safely assume the block is no longer represented in the local file. If we find a hit, we compute the strong checksum to verify. As noted previously, we use 160 bit SHA-1 for the strong checksum. If the strong checksum also matches, we conclude the remote block corresponds to the offset in the local file as indicated by the block we matched.

Tridgell used MD4 in rsync, but [Dobberton1, Dobberton2] points out attacks on MD4 such that a stronger hash is advisable when cryptographic strength is a goal. In rsync, the goal of the strong checksum is simple collision avoidance. In our case, though, we require our strong checksum to be able to certify the authenticity of a block as well as avoiding collisions.

After we have exhausted the remote blocks, we scan for holes in the covering and cover any such holes with new blocks. We send the new block list and the new blocks (all compressed and encrypted) to the remote and write a new summary signature in the local database. Every signature in the local database represents a full covering of the file.

A user more interested in conserving remote disk space than in an historical record of his files may delete those remote blocks and block lists which are no longer used and remove the old summary signature from the local database.

3.6 Cryptography

`cryptar` utilizes a standard block cipher for encrypting the data sent to the remote database. Currently the application uses the AES[FIPS-197] cipher, but the specific cipher is sufficiently abstracted in the implementation that another cipher could be substituted if it were desired, say due to the discovery of a vulnerability.

Since the `cryptar` blocks are significantly larger than the size of our cipher's block, we use the cipher in counter mode [Schneier]:

$$\begin{aligned} C_{-1} &= C_{-1} + 1 \pmod{2^W} \\ C_i &= P_i \oplus E_k(C_{-1}) \end{aligned}$$

The initial value of *C-1* is discussed below.

For reasons of security we can not use the cipher in electronic codebook (ECB) mode. If a single character is changed in a file and the resultant block is updated, the only difference between the initial `cryptar` block and the final `cryptar` block would be a single contiguous block of binary data whose length corresponds to the size of the block cipher used. This could form the basis of a differential cryptanalysis based upon a known ciphertext blocks and is therefore undesirable here.

Using counter mode requires us to use an initialization vector (IV) for each execution of our

cipher. We restrict each cipher instantiation to a single `cryptar` block, so each block is written to the data store with an individual, randomly generated IV. This value is stored as the first ciphertext block, *C-1*.

Counter mode alone provides weak security against someone changing a single cipher block. Assurance that the cipherblock has not been changed, however, is provided by the locally stored cryptographic (SHA-1) hash.

The storage of the IV in the clear on the data store does not decrease the security of our crypto scheme: the IV is cryptographically no different than another ciphertext block. Indeed, an additional benefit to rotating IVs on a per-block basis is that we are able to reuse a single key across many cryptar blocks without damaging the integrity of our key due to overuse.

It should be mentioned that while we are currently using only one key across the entire ciphertext, it is possible to rotate the key upon any schedule determined by the user, and this functionality has been implemented in the application. The use of a single key does not degrade security: the use of randomized initialization vectors for each block helps decrease the possibility of cryptanalyzing the key from the large ciphertext store.

We note, finally, that using a new random IV for each cryptar block that we encrypt prevents us from using a differencing scheme on the ciphertext blocks for cryptanalysis.

3.7 Hash Confidence

Because `cryptar` stores SHA-1 hashes for each block stored remotely, it has a very good assurance that the blocks it retrieves are the unmodified blocks that it originally transmitted [FIPS-180-1].

By way of illustrative example, consider a 10 MB file stored in 10,240 blocks of size 1K each. From a strictly probabilistic viewpoint, since SHA-1 is a 160 bit hash, the probability that no bad block passes the hash is

$$\begin{aligned}\left(1-2^{-160}\right)^{10240} &\approx 1-10240\left(2^{-160}\right)\\ &\approx 1-2^{13}2^{-160}\\ &= 1-2^{-147}\end{aligned}$$

More expansively, suppose we transfer 1000 such files per second without stop for 100 years. Then the number of blocks transferred would be

$$(1000)(10240)(60\cdot 60\cdot 24\cdot 365.25\cdot 100) \approx 2^{55}$$

and the probability that no bad block will pass the hash at some point during that century of work is

$$\begin{aligned}\left(1-2^{-160}\right)^{\left(2^{55}\right)} &\approx 1-2^{55}2^{-160}\\ &= 1-2^{-105}\end{aligned}$$

Thus, the probability of a bad block being mistakenly accepted for a good block during a 100 years of such transfers is 2-105.

If a bad block does pass the hash, it would still have to meet the file level hash, also a 160 bit SHA-1. Clearly the a priori probability of failure is extremely low.

It appears, moreover, that it is extremely difficult to create blocks that meet a given SHA-1 hash, let alone a block of a fixed length that must then meet a second (whole-file) SHA-1 hash.

Note that the guarantee we provide is that if we get data back, we will know that it is our data. We have no assurance that our data will not be deleted due to malice, disaster, or administrative error.

4 Security

The ready availability of ssh relieves us of any requirement to provide a secure protocol for communication with the archive. We simply have cryptar talk to the remote using the user's choice of secure transport agent.

We compress and then encrypt data before sending it to the archive. We allow for padding the encrypted data with random data to hide compression efficiency. We currently use AES[FIPS-197] to encrypt data.

We distinguish between two types of attackers: a network attacker and a remote host attacker.
The network attacker can see trafifc, but, presumably, make no sense of it, as the channel is encrypted by the prudent user. Some information is still available to the network attacker, however, such as the time of day at which backups are made and how much data is transferred.

As a fanciful example, if the amount of data tends to increment in units of 60 MB and the cryptar user is known to be fond of ripping CD's, the attacker might guess that the data represents mp3's encoded at 128 kbps.

A remote host attacker, on the other hand, could also glean some information from noting the order in which blocks have been sent to the archive. Although we could randomly generate remote block identifiers to hide block order, a remote attacker could still see the order in which the blocks arrive.

To impede such an attacker, we could transmit items randomly from our queue of items to send, even operating on more than one file at once, in an effort to hide information about which blocks come from which files. The protection thus gained, however, would not be significant compared to the protection afforded by AES itself.

In any case, if I know that you modified file foo this morning, and you do a backup this afternoon, and I am a sufficiently privileged attacker, I could at least learn that some subset of a small set of blocks corresponds to the delta of file foo.
In addition, because block lists are stored remotely, a remote host attacker could conclude that any session in which significant data moves from the local to the remote is an archive operation, and so those blocks that travel in the reverse direction are block lists.

It is at this point that we depend upon the strength of our encryption to protect our data. We use different random initialization vectors for each block, and we allow, moreover, for different encryption keys for different blocks. If multiple keys are utilized, an attacker with access only to the remote store and who successfully cryptanalyzes a single key would still not necessarily be able to gain access to a complete file, since a single file may require multiple blocks for a covering.

As previously discussed, the SHA-1 hash provides a very good assurance that the block we retrieve is, in fact, the block we sent[FIPS-180-1].

Finally, it should be mentioned that `cryptar` does not attempt to be robust against timing analysis.

5 Protocol and Usage

`cryptar` uses a simple binary wire protocol. We thought of using a popular protocol like XML or HTTP, but these protocols require encoding binary data as text, which introduces too much overhead. Since the bulk of the data transferred by cryptar is binary (compressed blocks), it made sense for the protocol to allow binary data without further encoding.

Each `cryptar` exchange consists of a one byte command, a version number (not shown below), and a payload. The commands are Hello, Bye, PutBlock, PutBlockAck, GetBlock, and GetBlockAck. The only version of the protocol for now is version zero.

(Hello, Message)
(Bye, Reason)
(PutBlock, id , data length , data)
(PutBlockAck, id)
(GetBlock, id)
(GetBlockAck, id , data length , data)

The interface to `cryptar` is similar to that of tar.

6 Performance

A frequent objection to rsync-like protocols is that a single changed byte invalidates an entire block. Indeed, an adversary (were our expected local environment adversarial) could change one byte every $L - 1$ bytes, where L is the block size, and thus invalidate every block. While it is important to note that the worst case performance of our algorithm deteriorates to full file copy, real world usage does not typically involve such regular and sparse changes.

7 Future Directions

The current `cryptar` implementation does not adequately address failure modes. If the network connection dies during backup, for example, remote blocks involved in the currently archiving file may be orphaned. This is a matter of bookkeeping that should be addressed. While important, its effect is limited to inefficiency: orphaned blocks merely waste space.

The protocol should be expanded to provide for statistically verifying the remote archive, as well as for verifying that a given file of the user's choice can be restored without necessarily restoring it.

If the user loses his entire local database, the current implementation of `cryptar` provides no mechanism for restoring anything or even notifying the remote host that the remote database is no longer needed. The latter is trivially addressed by the addition of a purge command to the protocol. The former is initially addressable through ordinary backup techniques. For example, while it is often impractical to back up a user's entire file set to removable media, the local database should fit on a CD-R.

This said, `cryptar` could recursively back up its own local database. Indeed, the entire local database could be encrypted and sent as a single block to the remote store. `cryptar` could then print a short bit of hex data representing the remote host, remote store identifier, and the block number, SHA-1, and encryption key to retrieve the local database. The user could, in the worst case, enter this information by hand to bootstrap a complete restore in the case of a total loss.

The protocol includes some bookkeeping information in order to match requests and acknowledgments. A clever attacker could, without breaking encryption, glean some information about which blocks go with which files, even though they would not know the names or contents of those files. Together with knowledge of the user's files, however, this could allow an attacker to focus his efforts more efficiently.

The protocol should be changed to avoid these revelations. The client would need to do more bookkeeping to handle request acknowledgment matching.

`cryptar` does not currently address the question of server reliability, only of data privacy. Using error correcting codes, one could distribute the data over several remote servers so that any 4 of them could fail without our losing data.

8 History and Status

`cryptar` is new software that borrows many ideas and some code from rsync. It is written in C, utilizes the GLIB and LIBTOMCRYPT [StDenis] libraries, and is available under the GPL. The initial implementation was built under Debian GNU/linux. While the software will no doubt grow to meet its users' extended needs, the core functionality described in this paper is implemented.

9 Conclusion

`cryptar` implements a scheme for securely storing and efficiently updating data on untrusted remote servers. As such, it provides an efficient and safe means to manage remote backups.

10 Availability

`cryptar` is available at http://www.cs.drexel.edu/~jeffa/cryptar/.

References

[Deutsch] P. Deutsch and J.-L. Gailly, RFC 1950:
ZLIB Compressed Data Format Specification version 3.3, http://www.ietf.org/rfc/rfc1950.txt (May 1996).

[Dobberton1] Hans Dobbertin, Cryptanalysis of MD5 Compress, (May 1996). http://citeseer.nj.nec.com/dobbertin96cryptanalysis.html

[Dobberton2] Hans Dobbertin. The First Two Rounds
of MD4 are Not One-Way. In Fast Software Encryption '98, pp. 284-292. Springer-Verlag, (1998). http://citeseer.nj.nec.com/dobbertin97first.html

[Rubel] Mike Rubel, Easy Automated Snapshot-Style Backups with Linux and Rsync, http://www.mikerubel.org/computers/rsync_snapshots/ (2002).

[Tridgell1] Andrew Tridgell and Paul Mackerras , *The rsync algorithm* http://samba.anu.edu.au/rsync/tech_report/tech_report.html, (November 1998).

[Tridgell2] Andrew Tridgell, *Efficient Algorithms for Sorting and Synchronization*, PhD Thesis, The Australian National University (April 2000).

[FIPS-180-1] Secure Hash Standard, Federal Information
Processing Standards Publication 180-1 http://www.itl.nist.gov/fipspubs/fip180-1.htm, April 1995.

[FIPS-197] Advanced Encryption Standard, Federal Information
Processing Standards Publication 197, November 2001.

[Quinlan] Sean Quinlan and Sean Dorward, *Venti: a New Approach to Archival Storage*, FAST 2002 Conference on File and Storage Technologies, (January 2002).

[Schneier] Bruce Schneier, *Applied Cryptography, 2nd Edition*,Wiley and Sons, Inc., 1996.

[StDenis] Tom St. Denis, LibTomCrypt, http://libtomcrypt.org/.

Formandens side

DKUUG skal fremad og op i gear!

Jeg er gået ind i DKUUGs bestyrelse sidste efterår og er nu fungerende bestyrelsesformand i foreningen.
Dette kommer efter at have arbejdet sammen med DKUUG i mange år med basis i SSLUG. I mange år har DKUUG støttet op om frivilligt arbejde indenfor open source-området og bl.a. gjort det muligt at afholde LinuxForum, Guadec II, Uptime(1) og mange andre events. Samarbejde er en af DKUUGs styrker.

DKUUG har fordele ved at have en god økonomi, basen i Symbion med mulighed for lokaler til møder og DKUUGs sekretariat som yder en solid indsats for at lave aktiviteter enten i DKUUG-regi, eller ofte sammen med foreninger såsom SSLUG, Perl Mongers og BSD-DK. Det er fordele jeg mener vi skal bruge aktivt, så DKUUG fortsat kan tilbyde medlemmerne i foreningen et varieret udbud af aktiviteter indenfor UNIX, Linux, BSD-familien, open source og åbne standarder.

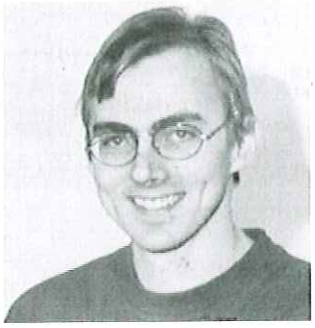
DKUUG har altid haft fokus på standardisering af åbne systemer, i høj grad via Keld Simonsens "livsmission" med at deltage aktivt i forskellige standardiseringsorganer. Det er efter min mening væsentligt at deltage i standardiseringsarbejde og at bidrage aktivt i internationale og nationale diskussioner om IT i fremtiden.
På det seneste har fokus igen været på åbne standarder, som er et "nyt" modeord i mange IT-sektorer. Det er kommet på mode at man støtter åbne standarder, men det har vist sig at nogle firmaer og organisationer aktivt prøver at bøje hvad der ligger i at være "åben". DKUUG har sammen med en lang række beslægtede organisationer lavet en definition af en åben standard, så det er klart hvad vi mener, når vi diskuterer åbne standarder. Dette er nærmere omtalt i en separat artikel her i bladet.

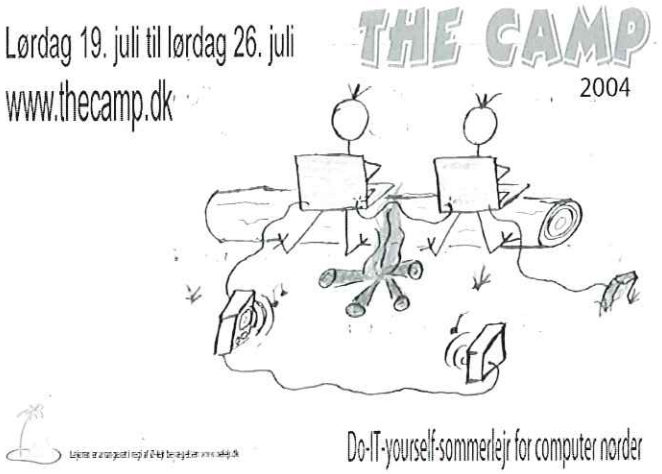
På de indre linier i DKUUG, så mener jeg at vi skal have set på at gøre det nemmere at deltage i foreningens arbejde og at få sikret en god balance af indflydelse til vores stormedlemmer, vores organisationsmedlemmer, vores individuelle medlemmer samt studiemedlemmer. DKUUGs bestyrelse vil bl.a. arbejde med dette spørgsmål de næste par måneder.

I er meget velkomne til at kontakte mig hvis I har input til hvordan I mener DKUUG skal arbejde i fremtiden.

Endeligt er det væsentligt at foreningens medlemmer er tilmeldt DKUUGs aktivitetsliste - se http://www.dkuug.dk/content/view/87/26/, da Post Danmark har forhøjet taksterne for udbringning af DKUUG-nyt væsentligt.
Det er muligt at DKUUG-nyt skal udkomme få gange per år, og til gengæld komme elektronisk. Nyheder om dette bliver omtalt i foreningens nyhedsbrev, som kommer til aktivitetslisten i starten af hver måned.

Peter Toft <pto@dkuug.dk>
Næstformand og fungerende formand DKUUG





For 3. gang præsenterer vi en do-IT-yourself-sommerlejr, for store og små computernørder. Vi har rammerne, en hytte med internetforbindelse, strøm nok og en hel uge til at lave præcist hvad vi har lyst til.
Lejren foregår i Jyderup på Bregninge gl. skole der idag kaldes Græsrodsgården fra lørdag d. 17 juli - lørdag d. 24 juli 2004 (uge 30). Billetter kan købes ved BilletNET der er et link til billetnet på lejrens hjemmeside http://www.thecamp.dk

Deltagere med billet har sikret en plads, hvis der er ledige pladser når lejren starter kan man dukke op og komme med, indtil lejren er fuld.

Prisen der inkluderer fuld kost og logi i 7 dage er for voksne (25år og derover) er 980 kr , unge (14-24år) betaler DKr 700,-
Herudover betaler alle kr. 50 kr. i infrastrukturbidrag.

Der er plads til 40 deltagere
Der er madrasser, man skal selv medbringe lagen og sovepose eller dyne.

Der vil blive holdt oplæg indenfor emner som Opensource operativsystemerne FreeBSD/Linux, netværk og om internet.
Niveaumæssigt er emnerne fra begynderniveau til øvede.

Vi forventer der bliver foredrag om følgende emner (dagligt kl 13.00):
FreeBSD kernen for begyndere og øvede: Poul-Henning Kamp (The FreeBSD project)
Netværksovervågning SNMP: Nicolai Petri (Catpipe)
Voice over IP: Jesper Rønnekilde, Robert Jeppesen (Tele Greenland)
Embedded Computing (FreeBSD/Soekris): Henrik Kramshøj (BSD-DK)
GRID computing og Clustering: Poul Erik Thamdrup, Bernd Dammann. (DTU)
Installation af FreeBSD: Sven Esbjerg, Kristen Nielsen (BSD-DK, DKUUG)

Vi har også mindre workshops/BOFs (om aftenen) om følgende emner (flere kommer løbende til, også under selve lejren)
TCP/ IP-netværk for begyndere Kristen Nielsen
Backup og Backup løsninger: (flere oplægsholdere)

Herudover er der naturligvis tid til hygge både foran computeren, ved lejrbålet, når vi laver og spiser mad osv.
Lejren arrangeres af ølejrbevægelsen, DKUUG yder vigtig støtte for at lejren kan afholdes, bla. til etablering af netforbindelse mv.

Åbne standarder

DKUUG, SSLUG, PROSA, IT-Pol, OSL, Internet Society (DK), BSD-DK, Digital Forbruger Danmark og KLID er gået sammen om en fælles forståelse og definition af en åben standard. Arbejdet med at lave en fælles definition har været drevet igennem DKUUGs standardiseringsgruppe STD med deltagelse af en række repræsentanter fra de ovennævnte grupper.

Definition af en åben standard

Version 1.1 - 2004-05-04

Formål

Formålet med åbne standarder er at sikre "interoperabilitet" (at systemer kan samarbejde problemfrit) mellem uafhængige produkter. Dette sikrer fri konkurrence på et frit marked og gør det samtidig betydeligt nemmere for brugerne af produkterne at anvende dem i nye sammenhænge på nye måder til gavn for samfundet, som derved får fuld udnyttelse af de aktuelle teknologier.

Det er derfor nødvendigt at blive enige om en fælles definition af åbne standarder, der kan anvendes af alle.

Definition

En åben standard opfylder følgende krav:

1: Veldokumenteret med den fuldstændige specifikation offentligt tilgængelig.
2: Frit implementerbar uden økonomiske, politiske eller juridiske begrænsninger på implementation og anvendelse.
3: Standardiseret og vedligeholdt i et åbent forum (en såkaldt "standardiseringsorganisation") via en åben proces.

Uddybning af definition på åbne standarder

Ad 1. Veldokumenteret

Et produkt, der hævder at basere sig på åbne standarder, skal for alle snitflader der er nødvendige for interoperabilitet og for at kunne integreres i og interagere med omverdenen, kunne henvise til den fuldstændige tekniske dokumentation, der er nødvendig for at skabe interoperable produkter.

Det er et krav for at sikre denne interoperabilitet og for at mindske mulighederne for tekniske problemer i interaktionen mellem forskellige produkter, at standardens præcise navn/ betegnelse samt fuldstændige og udførlige dokumentation er let tilgængelig. Dokumentationen skal være offentligjort på almindelig vis.

Kravet tilgodeser, at det er teknisk muligt at implementere standarden med den maksimale grad af interoperabilitet, og at alle der er interesserede i at implementere standarden uhindret kan de få fat i standarden.

Ad 2. Frit implementerbar

Der må ikke følge nogen bånd med på anvendelse og implementation af standarden. Det må f.eks. ikke være forbundet med afgifter at anvende den eller distribuere et produkt, der implementerer den. Der må ikke være politiske begrænsninger som f.eks. diskrimination overfor nationalitet, køn, race eller religion på anvendelse eller implementation. Anvenderen må ikke skulle forpligte sig ved indgåelse af specifikke juridiske aftaler/kontrakter for at kunne anvende eller implementere standarden.

Kravet tilgodeser at ingen udelukkes fra samfundets infrastruktur for eksempel p.g.a. økonomiske, kulturelle eller politiske forhold og at ingen producent udelukkes fra at lade sine produkter være anvendelige i infrastrukuren.

Ad 3. Vedligeholdt i åbent forum i en åben proces

En åben proces forudsætter at der foreligger klare, gennemskuelige regler for standardiseringsproceduren.

Det er vigtigt for en åben standard at den er uafhængig af individuelle interesser og at alle har mulighed for at bidrage til dens vedligeholdelse. Dette sikrer både standardens integritet, dens anvendelighed og internationale anerkendelse.

Kravet tilgodeser at enhver kan have indflydelse på standardens indhold. Det er væsentlig at standardens tekniske udformning er, så alle brugeres behov bliver varetaget. F.eks. er det væsentligt for danske brugere, at POSIX-standarden indeholder understøttelse for internationalisering og dermed mulighed for understøttelse af dansk (localer), at der findes udvekslingsformater for email der er 8-bit (ESMTP), at der findes en localestandard der bl.a. kan definere at ugen starter på en mandag, samt hvornår uge 1 er i et år (ISO TR 14652), at der findes en sorteringsstandard (ISO 14651), og at der findes tegnsætskonverteringsstandarder (som RFC 1345). Det er her også vigtigt at processen er virkeligt åben, så selv små aktører kan have indflydelse. Ingen af de ovennævnte faciliteter ville formentlig have være standardiseret, hvis små danske aktører ikke havde haft en reel stemmeret i det pågældende standardiseringsorgan. Det er svært at vide om disse facililteter overhovedet ville være blevet standardiseret, men sikkert er det, at de ikke ville have været nær så udbredte.

Denne definition er udarbejdet og vedligeholdes i DKUUGs standardiseringsudvalg <std@dkuug.dk>, der et åbent udvalg under DKUUG.

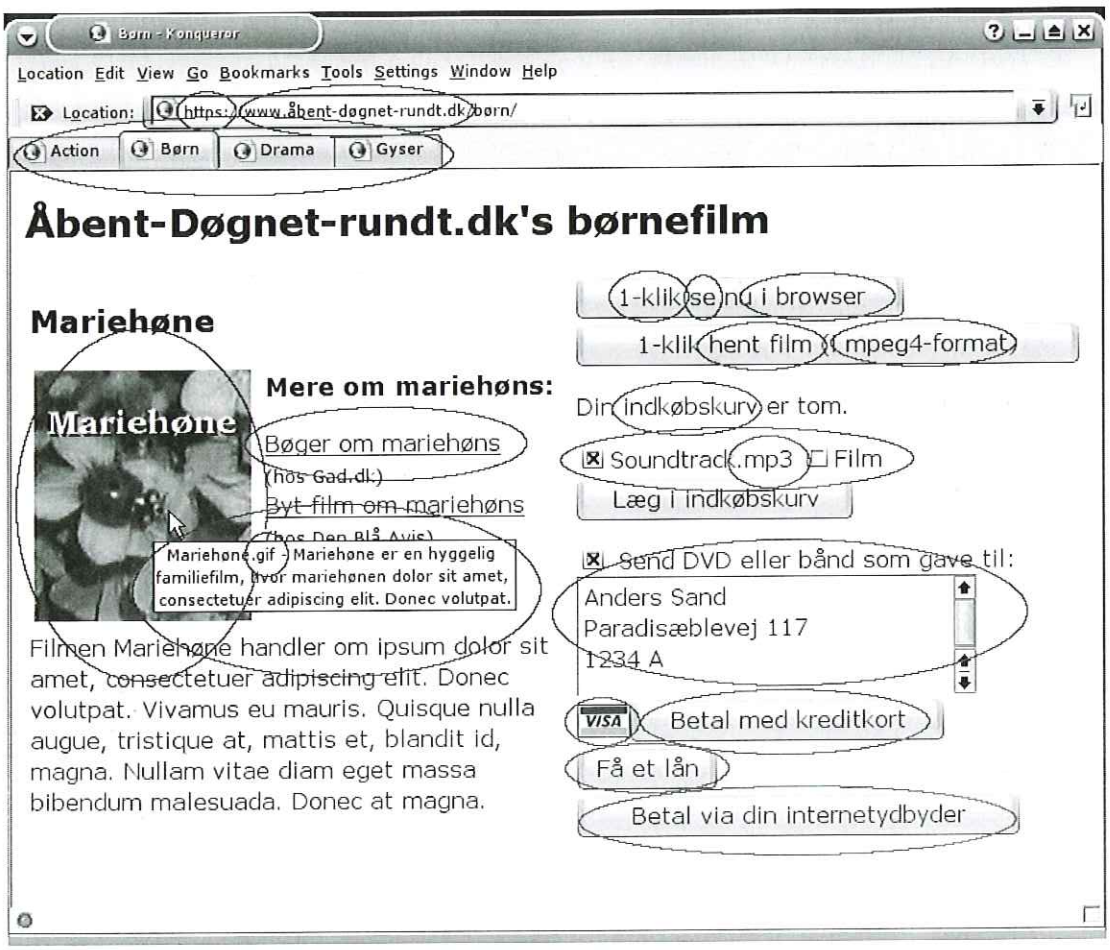
Softwarepatenter - det berører også din virksomhed

Af Ole Tange <tange@it-pol.dk>, best.medlem i IT-Politisk forening.

Danske video-butikker på nettet er for nyligt blevet anklaget for patentkrænkelse og skal betale 2% af deres omsætning for at benytte ideer, som de selv har fundet på. Softwarepatenter begrænser sig ikke til videobutikker, men spreder sig ind i alle brancher, der benytter IT.

De fleste virksomheder føler sig som lovlydige borgere og mener ikke, at de krænker andres patenter. Men i modsætning til uautoriseret kopiering så kan man ikke selv vide, om man krænker patenter. En simpel ide, som man selv har fået, kan en anden have patenteret. Og inden for software sker dette ofte. Derfor kommer softwarepatenter ofte som en overraskelse for virksomhederne.

På billedet er skitseret en videobutik, hvor en stribe af funktionerne viser sig at være patenterede af andre end videobutikudvikleren.



- åbent-døgnet-rundt.dk - ÆØÅ i domænenavne: EP1159820
- [Drama] [Børn] [Action] [Gyser] - Faneblade: EP689133
- Mariehøne er en hyggelig... - Pop-up vindue: EP0537100
- Se - Videovisning via web: EP0933892
- Hent film - Videovisning via web (samme som ovenfor): EP0933892
- mpeg4-format - Udbredt videoformat til download af video: Mere end 40 patenter (herunder DK638218)
- Indkøbskurv - Konceptet indkøbskurv: EP0784279
- Soundtrack.mp3 og Film - E-handel med serviceydelser: DK/EP738446
- Soundtrack.mp3 - MP3 er et udbredt lydformat: DK287578
- Betal med kreditkort - Kreditkortbetaling via internettet: EP820620
- Send som gave - Beregning af korrekt gaveadresse ud fra mangelfulde oplysninger: EP927945
- Få et lån - Automatisk lån af penge: EP715740
- VISA - Digital signatur i grafikfil til at checke at det er godkendt til at modtage VISA-kort: EP0798657
- Betal via din internetudbyder - Betaling via et telekommunikationsnetværk: EP084836
- 1-klik - Køb ved hjælp af et klik: US5960411 (Europæisk ansøgning: EP0902381)
- https:// - SSL-forbindelser benytter krypterede sockets: US5657390
- i browser - Visning i et vindue i en browser: US5838906
- Bøger om mariehøns - Link til samarbejdspartner: US6029141
- Mariehøne.gif - GIF-filer benytter LZW-komprimering: EP129439
- https:// - SSL-forbindelser benytter RSA-kryptering: US4405829

Nogle af patenterne er gyldige i Danmark, andre i Europa eller USA og atter andre er udløbet. Fælles for dem er, at kender man ikke til deres eksistens og deres legale status, så risikerer man en ubehagelig overraskelse i form af et krav om patentlicens.

Problemet er kommet til Danmark.

Hvor risiko for licenskravet tidligere har været lille og reelt begrænset sig til udenlandske virksomheder, så er risikoen pludselig blevet helt reelt for danske virksomheder. Danske video-butikker på nettet er for nyligt blevet mødt af et krav for krænkelse af patentet EP0933892.

EP0933892 er et patent ejet af det amerikansk selskab Acacia. Patentet er gyldigt i Danmark og dækker både hentning (download) og visning (streaming) af video via internettet.

Video-butikkerne er blevet afkrævet 2% af omsætningen i licens for brug af en ide, som butikkerne har fået uafhængigt af Acacia. Acacia laver ikke produkter selv, men er istedet en patentfabrik, som ejer en stribe patenter inden for IT, media og biotek.

"Det holder ikke i retten".

Der er udstedt en stribe patenter på ideer, som selv patentadvokater indrømmer ikke opfylder kravene til en opfindelse. Det er muligt at underkende disse, men i praksis sker det ikke. Forklaringen er enkel: Modtager en virksomhed et krav som følge af krænkelse af et tvivlsomt patent, så vil de fleste virksomheder foretage en forretningsmæssig vurdering af, om man vil prøve at underkende patentet eller man blot vil betale. Vælger man ikke at betale, så risikerer man at få et fogedforbud, der gør at man ikke må forsætte krænkelsen, før sagen er afgjort. Det

kan f.eks. betyde at man ikke må sælge sine produkter eller serviceydelser.
Man pålægger også virksomheden den byrde at blive trukket ind i en patentretssag - en sådan sag koster 1-5 mio kr, og der er altid en risiko for at tabe.

Hvis valget står mellem en patentretssag til 1-5 mio kr. og en licens for det tvivlsomme patent på kr. 100-500.000 så er valget for de fleste virksomheder ret enkelt.

Lovgivningen.

I den danske patentlov står i §1: "Som opfindelser anses især ikke hvad der alene udgør [...] programmer for datamaskiner".

Man kan derfor undre sig over, at man kan fortolke loven så man tillader patenter på faneblade (EP689133), pop-up vinduer (EP0537100) og kreditkortbetaling via Internettet (EP820620). Og det er da så absolut muligt, at patentmyndighedernes praksis er i strid med loven.
EU-kommissionen har derfor fremsat et direktiv, som klart vil lovliggøre den tvivlsomme praksis. I EU-parlamentet er direktivet blevet ændret, så det kun tillader patenter på software, der direkte påvirker hardware. I praksis vil parlamentets ændrede direktiv ugyldiggøre alle patenterne, som rammer video-butikken.

Den endelige vedtagelse mangler endnu, og direktivet kan nå at blive ændret mange gange inden vedtagelsen. Der er derfor rig mulighed for at påvirke direktivet. Det er patentindustrien og patentstyrelserne naturligvis klar over, og de prøver af magt at påvirke politikerne til at udvide deres forretningsområde.

Påvirk loven.

Hvis din virksomhed benytter bare en smule IT, så er sandsynligheden høj for, at I (ligesom videobutikken) krænker et eller flere softwarepatenter.
Du bør derfor tale med din chef om hvilke forholdsregler I vil tage. Hvis I kommer frem til, at softwarepatenter ikke er til gavn for jeres virksomhed, så vil IT-Politisk forening meget gerne høre fra jer.

Hvorfor er åbne standarder vigtige?

Formålet med åbne standarder er at sikre "interoperabilitet" (at systemer kan samarbejde problemfrit) og fri konkurrence på markedet mellem uafhængige produkter.

Fri konkurrence
Et frit marked og gør det samtidig betydeligt nemmere for brugerne af produkterne at anvende produkterne i nye sammenhænge på nye måder til gavn for samfundet, som derved får fuldt udnyttelse af de aktuelle teknologier.

Begrebet
"Standard" er et ord man bruger mange steder. Hvis man f.eks. "plejer" at anvende et bestemt produkt i en branche/firma kan man sige at der er tale om en "de facto standard".

De standarder vi interesserer os for her er tekniske specifikationer udfærdiget eller godkendt i et standardiseringsorgan. Denne tekst handler om standarder inden for IT-verdenen og specifikt det man kalder "åbne standarder". Normalt er der tale om "protokoller", "formater" eller "interfaces".

Hvorfor er åbne standarder vigtige indenfor IT-verdenen? Dette kan det være teknisk kompliseret at sætte sig ind i for ikke-fagfolk.
For at se nødvendigheden er det vigtigt at huske på at Internettet er blevet en vigtig del af vores globale infrastruktur. En stor del af al kommunikation foregår i dag over Internettet og mængden af denne kommunikation bliver større for hver dag.

Selv for 2-3000 år siden anerkendte man vigtigheden af at vi kunne kommunikere og forstå hinanden. Den bibelske beretning om Babelstårnet (Tårnet blev ifølge 1. Mosebog 11, 1-9, bygget for at nå himlen; Gud straffede da mennesket ved splittelser og sprogforvirring) er netop et eksempel på hvor vigtigt det er for folk at kunne kommunikere.

Du kender sikkert selv fornemmelsen af at stå i et fremmed land uden at kunne sproget. Tænk hvis vi end ikke havde mulighed for at lære andres sprog?

Hvis sproget var en forretningshemmelighed, eller blot forbudt at tale uden tilladelse fra myndighederne? Hvordan ville du f.eks. kunne slå dig ned som handlende i et fremmed land, hvis myndighederne holdt sproget hemmeligt for udlændinge?

Man kan ikke basere en fælles samfundsmæssig infrastruktur på kommunikationsformer, der er ikke er frit tilgængelige for alle.

I forhold til mennesker er computere meget lidt tilgivende. Hvis to systemer skal kunne tale sammen skal de tale præcist samme sprog. Der er ikke plads til dialekter eller slang. Derfor er det uhyre vigtigt at producenter af hardware og software kan få adgang til de fuldstændige tekniske specifikationer af hvordan andres produkter kommunikerer, ellers kan deres egne produkter ikke interagere og integreres i andre produkter på markedet. Kan de ikke det, har de ikke en chance blandt store udbydere på markedet.
Der er ikke tale om, at producenten skal forære hele resultatet af sin produktudvikling til almenheden. Hvordan produktet er lavet, er ikke det væsentlige. Enhver producent er selvfølgelig velkommen til at beholde sin produktudvikling som en forretningshemmelighed.

Det, der er vigtigt, er oplysningerne om hvordan andre aktører i markedet kan kommunikere med produktet.
Hvis ikke disse oplysninger er baseret på åbne standarder, bør samfundet ikke basere sin digitale infrastruktur på dem.

Verdensrekord i hus

Fredag d. 30/1 var DKUUG vært for et verdensrekordforsøg i distribueret skak....og knap 4 timer senere var verdensrekorden i hus: 2070 computere fra over 56 lande havde deltaget i rekordforsøget.

I begyndelsen havde ChessBrains centrale server - den Linux baserede server applikation - SuperNoden store vanskeligheder, grundet det store antal af klienter (PeerNodes) der havde tilmeldt sig. Det er SuperNoden, der bl.a. står for at schedulere og uddele jobs til klienterne og den blev på det nærmeste DDoS'et, da alle klienterne meldte sig til næsten samtidig. De fremmødte skakfolk fnisede lidt i krogene, da ChessBrain efter 15 minutter fortsat ikke havde lavet et modtræk til Peter Heines åbningstræk.

Efter et stykke intenst arbejde fra ChessBrain-folkene formåede de at få lavet et program, der i real-time modificerede SuperNodens /proc filsystem og sorterede klienterne efter styrke og alle de "dårligste" klienter blev smidt af serveren. Dette tillod klienter med mest CPU-kraft og hukommelse at komme til og efter 40 minutter kom spillet rigtigt i gang.

ChessBrain-folkene havde ellers arbejdet i døgndrift op til selve arrangementet, for at få stresstestet deres applikation igennem og de sad fysisk placeret i rummet ved siden af skakspillerne.

Spillet blev fulgt at en stor folk skakentusiaster samt open source folk, som sad i Symbions M1 og fulgte slagets gang via storskærm. Skærmen var delt op, så et billede viste spillerne og et andet billede viste en grafisk repræsentation af skakbrættet.

Overfor Peter Heine Nielsen sad Peter Wilson og udførte computerens træk, på et fysisk skakbræt. Peter Wilson var tilstede som repræsentant for den internationale skak-union og som repræsentant for Guinness Book of World Records. Peter Wilson var formand for den internationale skak-union fra 1998-2002 og sad i den ekspert komite i New York, der i 2003 overvågede skakspillet mellem verdensmesteren i skak Garry Kasparov og IBM computeren Blue Junior. Det var derfor ikke noget nyt for ham, at se et menneske spille overfor en computer.

ChessBrain systemet er i sandhed et distribueret system. Her er klienterne ikke begrænset til en klynge af ens maskiner, men kører på en skønsom blanding af servere, workstations, bærbare og folks hjemmepc'ere. ChessBrain har lavet klienter til Microsoft Windows, Linux, FreeBSD og Mac OS X. Linux Journal bragte i sin September 2003 udgave en grundig gennemgang af arkitekturen og designfilosofien bag ChessBrain (se nedenstående links).

Kl. 22:57 var spillet slut. Peter Heine tilbød, som en ren gentleman, at det blev remis, da computeret havde 45 sekunders spilletid tilbage. Dette på trods af at Peter Heine havde computeren i skak.

En stor tak til alle, der deltog med computerkraft. Det er iøvrigt nu muligt at hente et certifikat fra ChessBrains hjemmeside, som viser, at man har deltaget med computerkraft i verdensrekordforsøget.

Verdensrekorden kommer i 2005 udgaven af Guinsess Book of World Records under titlen : "Largest networked chess computer".

Det kan tænkes at ChessBrain-folkene vender frygteligt tilbage til Danmark næste år, da de i den grad gerne vil have en rematch med Peter Heine Nielsen og de dermed har haft yderligere 1 år til at forbedre ChessBrain.
Der arbejdes i øjeblikket aktivt på at få stablet en rematch på benene f.eks. til LinuxForum 2005.

Links:

Du kan se streaming fra arrangementet på adressen:
http://dkuug.mmmanager.net/chessevent.smil

http://www.chessbrain.net/hdcp.html
http://www.chessbrain.net/wra/
http://www.linuxjournal.com/article.php?sid=6929
http://www.linuxjournal.com/article.php?sid=7408
http://www.newsforge.com/software/04/02/11/2057247.shtml

Foto: Sidsel Jensen

Nyhedsbrev - maj

Nyhedsbrev fra DKUUG - hvad er det?

For at give bedre synlighed på det arbejde som sker i DKUUG, så er det planen at lave et månedligt nyhedsbrev som kan skabe bedre kontakt fra bestyrelsen til medlemmerne og give bedre overblik over hvad der sker i udvalgene m.m.

Bestyr - bestyrelsen i DKUUG

Hjemmeside : http://www.dkuug.dk/content/view/10/27/

Strategi-weekend

Bestyrelsen arbejder på at få afholdt en strategi-weekend, hvor bestyrelsen kan få afstemt sine langsigtede mål og visioner for DKUUG og sine kortsigtede mål for bestyrelsesperioden.

Styrket samarbejde med open source foreninger

DKUUG ønsker at støtte open source arbejdet i Danmark og afholdt i den forbindelse et møde tirsdag d 27/4 kl. 17 på Symbion, hvor SSLUG og BSD-DK mødte op og gav deres besyv med, i forhold til hvordan samarbejdet skal foregå.
Grundideen er, at de enkelte foreninger kommer til at stå langt stærkere hvis der samarbejdes på tværs.

KLB - Klub-møder

DKUUGs møder afholdes sidste tirsdag i hver måned kl. 19.
KLUB-udvalget i DKUUG står for at arrangere forskellige tekniske klub-aftener for DKUUGs medlemmer. Hvis du har en ide til et spændende teknisk emne eller gerne vil være med til at arrangere disse aftener, så send en mail til <klb@dkuug.dk>
Næste KLB møde er tirsdag d. 25/5 og mødeansvarlig er Peter Toft.

3 konferencer i 1.kvartal

2004 startede lige på og hårdt for DKUUG var medarrangør af ikke mindre end 3 konferencer i starten af året: NordU, LinuxForum samt Nordic Perl Workshop.

Skak-verdensrekord i hus

I forlængelse af NordU, blev der fredag aften afholdt et Guiness Book of World Records - verdensrekordforsøg i distribueret skak. Partiet blev gennemført og spillet mod den danske stormester Peter Heine Nielsen resulterede i uafgjort. Peter Heine spillede imod den kombinerede regnekraft fra 2,070 computers, som kom fra over 56 forskellige lande.

Du kan læse Carlos Justianos beskrivelse af arrangementet på ChessBrains hjemmeside:
http://www.chessbrain.net/wra/cjrecount.html

Verdensrekorden er også blevet omtalt i maj-udgaven af LinuxJournal:
http://www.linuxjournal.com/article.php?sid=7408

Nyhedsbrev - maj

Du kan også se streaming fra arrangementet på adressen: http://dkuug.mmmanager.net/chessevent.smil

Hyggeligt klub-møde om FriFinans

I slutningen af februar blev der afholdt klub-møde om det helt nystartede open source regnskabsprojekt FriFinans (se www.frifinans.dk).
Torben Sørensen fra CasaLogic var initiativtager og godt 15 interesserede var mødt op for at høre hvor langt de var kommet og hvad FriFinans kunne, som f.eks. PC Plus og SQLedger ikke kunne.
FriFinans mangler stadig folk der gerne vil udvikle på projektet.

Shell Celebrity Deathmatch

I april blev der afholdt et fælles arrangement i M2 i Symbion, hvor DKUUG, SSLUG og BSD-DK lavede "Shell Celebrity Deathmatch".

Det blev en meget festlig aften, med en veloplagt trio med Sidsel Jensen, Hanne Vilmann, og Gitte Wange som über-dommere forsøgte at styre løjerne - seks talere var i hopla den aften:

* sh : Thomas Ammitzbøll-Bach (DKUUG)
* bash : Peter Makholm (SSLUG)
* ksh : Jesper Louis Andersen (BSD-DK)
* zsh : Henrik Christian Grove (SSLUG)
* rc : Anders Søndergaard Jensen (BSD-DK)
* cmd.exe : Eske Rahn forsvarede Windows cmd.exe :)

Pointgivning blev dels dikteret af publikum, samt chikane fra en taler mod andre talere - og fedteri for dommerpanelet. Det var morsomt :)

Jesper Louis Andersen løb med sejren - tillykke til ham og ksh.

Et par billeder kan findes på http://pto.linux.dk/shell - gå ikke glip af http://pto.linux.dk/shell/index.php?viewone=1

Gik man glip af aftenens arrangement, bliver slides og streaming af foredragene tilgængelig fra http://www.bsd-dk.dk/old_events.php

BLD - DKUUGs gruppe der skriver DKUUG-nyt

DKUUG-nyt er som så mange andre fagblade presset af at porto-tilskud er faldet bort. Hanne Vilmann har på vegne af DKUUG ansøgt om at komme ind i en tilskuds-pulje som kan hjælpe på DKUUG-nyts økonomi.
Men der er endnu ingen afklaring på hvorvidt vi er kommet med i denne pulje.

Vi opfordrer til at man får lavet nogle gode artikler og boganmeldelser til DKUUG-nyt - for en god boganmeldelse giver vi den bog du anmelder.

Ud over DKUUG-nyt i anledning af NordU 2004 og LinuxForum 2004, så er planen at udgive

Nyhedsbrev - maj

følgende numre i 2004.

DKUUG-nyt Nr 148: Deadline den 17/5 - Udsendes den 28/5

DKUUG-nyt Nr 149: Deadline den 28/8 - Udsendes den 13/9

DKUUG-nyt Nr 150: Deadline den 05/11 - Udsendes den 22/11

STD - DKUUGs standardiseringsliste

Hjemmeside : http://std.dkuug.dk

Keld Simonsen har været udsendt fra DKUUG til møde i ISO SC22 Linux Rapporteur group 2004-02-03/05 i Tokyo. Et af de store diskussionsemner fra afvejningen mellem standardisering af Linux via Free Standards Group http://www.freestandards.org/ og ISO. Ved samme lejlighed var der møde i openi18n-gruppen, hvortil Keld har indsendt bidrag med titlen "Linux in every corner of the world."

DKUUGs bestyrelse har desuden lavet en særbevilling til STD, så Keld Simonsen kunne deltage i ISO/IEC JTC1/SC34 møde i Amsterdam, 2004-04-14/20. SC34 er den underkomité som dækker dokumenthåndtering i ISO. De står bl.a. for iso-ficeringen af HTML, XML og docbook. Mere info på http://www.y12.doe.gov/sgml/sc34/

STD har igennem længere tid arbejdet med definitionen af en åben standard.
Peter Mogensen har det sidste halve år været den primære drivkraft bag skrivearbejdet, og resultatet findes på http://www.aaben-standard.dk/.
Planen er nu at samle de foreninger og firmaer, der har lyst til at støtte definitionen.

DKUUG definerer en åben standard ved:

---

* Veldokumenteret med den fuldstændige specifikation offentligt tilgængelig.
* Frit implementerbar uden økonomiske, politiske eller juridiske begrænsninger på implementation og anvendelse.
* Standardiseret og vedligeholde i et åbent forum (en såkaldt "standardiseringsorganisation") via en åben process.

DKUUGs standardiseringsgruppe STD har indsendt høringssvar om fremtidens IT-standarder. Det er pointeret at DKUUG støtter brugen af åbne standarder, og at vi gerne vil bidrage med hjælp til dette. Hele høringssvaret kan læses på http://std.dkuug.dk/dkuug/oio

STD er en åben gruppe som kan kontaktes på <std@dkuug.dk>.

NET - Netudvalget

Ny FTP server på vej

Pt. arbejder NET-udvalget på at få bygget en ny og mere tidssvarende FTP server (ftp.dkuug.dk). Den gamle FTP server (thot) har længe haft store problemer med manglende

Nyhedsbrev - maj

plads og selve maskinen var langsom. Der blev derfor i december indkøbt en ny maskine samt RAID, så der igen kan blive masser af plads (1.5TB) og de forskellige OS mirrors kan blive opdateret.

Hvis du har forslag til et mirror, du gerne så DKUUG havde, så send en mail til net@dkuug.dk>, så finder vi ud af om det kan lade sig gøre.

www.dkuug.dk - styrket webmaster hold

Sidste år blev DKUUGs hjemmeside langt om længe opdateret. Det havde været tiltrængt længe, men lykkedes endelig takket være Benny Kjærgaards indsats. Serveren blev lanceret i forbindelse med foreningens 20 års jubilæum i November og den benytter open source systemet MAMBO.

Der blev i den forbindelse nedsat et webmaster hold, som prøver at ajourføre hjemmesiden i langt højere grad end tidligere. Hvis du, som medlem kunne have lyst til at hjælpe til med denne indsats kan du skrive til <net@dkuug.dk> eller <webmaster@dkuug.dk>

Wireless udstyr på kontoret

DKUUGs NET-udvalg valgte i januar at indkøbe 2 stks Apple Airports, som blandt andet blev brugt under Perl Workshoppen i marts. Hvis du står og skal afholde et DKUUG/open source arrangement og har brug for wireless dækning, kan de to airports lånes ved forudindgået aftale med DKUUG.

Arbejdsweekender i NET

NET udvalget står foran at skulle opgradere en masse på maskinparken og afholdt derfor i slutningen af april en arbejdsweekend, hvor arbejdet blev påbegyndt.

Net afholder endnu en arbejdsdag i starten af maj, hvor der arbejdes videre på de påbegyndte projekter og andre opdateringer af drifts udstyr.

EXT - Externt udvalg.

Externt udvalg havde i starten af marts møde med den Norske NUUG, svenske Europen.se og finske FUUG unixbrugergruppe, omkring generelt samarbejde og fremtiden for NordU konferencen. Der blev nedsat et NordU advisory board med medlemmer fra alle landene, og det blev besluttet at Norge afholder NordU i 2005 i Oslo. Der er i øjeblikket arbejde i gang med at finde en programkomité til NordU, herefter starter arbejdet med den næste konference op. NUUG afventer med interesse evalueringen og resultaterne for NordU2004 fra DKUUG.

Kort nyt fra ext relationer:

---

* GUADEC (www.guadec.org) GNOME User And Developer European Conference i Kristianssand (28-30 juni) har startet registrering af gæster.
* SANE konferencen i Holland (www.sane.nl) 27 sep - 1 okt 2004 i Amsterdam, Holland, har udsendt call for papers and posters.
* BSDCON-Europe 29-31 oktober 2004, (http://2004.bsdconeurope.org) Karlsruhe, Tyskland, har udsendt call for papers.

Nyhedsbrev - maj

KLID

DKUUGs interessegruppe KLID har lavet en del ting i de forløbne måneder.

Vi har fået på oversat på Fedora og Mandrakes installationsprocedurer, og i vores projektorienterede ftp-service har vi fået en del nye ting ind. Det største hit er Mandrake 10.0 Community, som man ved at installere opdateringer faktisk allerede nu kan opgradere til en fuld Mandrake 10.0 Official. Så har vi fået en Fedora testudgave 1.92, og vi har etableret spejl for Skolelinux og tilhørende Debian. Sluttelig har vi etableret et spejl for SuSE, som vi også har været med til at oversætte.

Nye indholdsmæssige ting på websiderne, er en dokumentation på dansk af KDE, udarbejdet af Erik Kjær Pedersen. Også nyt er en dansk bog om OpenOffice.org, skrevet af Rolf Larsen og Henrik Just. Desuden har der været nogen opdateringer på vores vejledning om hvad man kan bruge af programmer i Linux i stedet for Windows-programmer, på den engelsk-danske ordliste og vores oversigt over økonomien i Linux.

Der er kommet lidt gang i vores nyhedstjeneste igen på klid.dk, efter en pause.

Der har været to medlemsmøder, et om Total Cost of Ownership og modeller herfor inden for det offentlige, med Michael Hald fra KL, og et møde i forbindelse med den ekstraordinære generalforsamlingen økonomisystemerne Hansa og det danske SQL Finans, med Peter Rude og Matthijn Buur fra ITz.
Begge møder var livlige med mange spørgsmål fra salen. Vi forsøger altid at lægge plancher fra foredragene op på web-siderne (under afholdte arrangementer) bagefter.

Vi har holdt generalforsamling og ekstraordinær generalforsamling, for at få vedtaget vedtægtsændringer så vores kommunikation med medlemmerne ikke behøver at gå via papirpost, men at vi kan bruge email. Vi har også fået to nye bestyrelsesmedlemmer, Delin Wei og Anders Kr. Andersen.

Opfølgning på nyhedsbrev og deltagelse i udvalg

Har du lyst at være med i DKUUGs udvalg, så skriv til det enkelte udvalg at du har lyst til at deltage.

DKUUG har også en debat-liste <debat@dkuug.dk> som er til diskussion
om foreningen og gerne dette nyhedsbrev.
Tilmelding til debat-listen sker via http://www.dkuug.dk/content/view/87/26/

Nyhedsbrev - juni

Nyhedsbrev fra DKUUG - hvad er det?

For at give bedre synlighed på det arbejde som sker i DKUUG, så er det planen at lave et månedligt nyhedsbrev som kan skabe bedre kontakt fra bestyrelsen til medlemmerne og give bedre overblik over hvad der sker i udvalgene m.m.

Bestyrelsen

På det ordinære bestyrelsesmøde 11. maj blev der ændret i bestyrelsessammensætningen. Benny Kjærgaard meddelte at han ønskede at udtræde af bestyrelsen, hvilket blev taget til efterretning. Sidsel Jensen overtager Bennys post som formand for ADM-udvalget. Formand for DKUUG Ulf Nielsen meddelte derefter at han af private årsager måtte bede om orlov på ubestemt tid fra DKUUGs bestyrelse. Ulf bad om at næstformand Peter Toft blev indsat som fungerende formand, hvilket bestyrelsen fulgte.

Bestyrelsen har desuden indkøbt et par domæner med navne, der navnemæssigt ligger tæt op af aaben-standard.dk.

Bestyrelsen planlægger at få afholdt ordinær generalforsamling 16/11 2004.

Kontakt : <bestyr@dkuug.dk>

The Camp

For 3. gang præsenterer vi en do-IT-yourself-sommerlejr, for store og små computernørder. Vi har rammerne, en hytte med internetforbindelse og strøm nok og en hel uge til at lave præcist hvad vi har lyst til. Lejren foregår i Jyderup på Bregninge gl. skole der i dag kaldes Græsrodsgården fra lørdag d. 17 juli - lørdag d. 24 juli 2004 (uge 30).
Billetter kan købes ved BilletNET der er et link på lejrens hjemmeside http://www.thecamp.dk

Der vil blive holdt oplæg indenfor emner som Open source operativsystemerne FreeBSD/Linux, netværk og om internet. Niveau-mæssigt er emnerne fra begynderniveau til øvede.

Vi forventer der bliver foredrag om følgende emner (dagligt kl 13.00):
FreeBSD kernen for begyndere og øvede:
* Poul-Henning Kamp (The FreeBSD project)
* Netværksovervågning SNMP: Nicolai Petri (Catpipe)
* Voice over IP: Jesper Rønnekilde, Robert Jeppesen (Tele Greenland)
* Embedded Computing (BSD/Soekris): Henrik Kramshøj (BSD-DK)
* GRID computing og Clustering: Poul Erik Thamdrup, Bernd Dammann. (DTU)
* Installation af FreeBSD: Sven Esbjerg, Kristen Nielsen (BSD-DK, DKUUG)

Vi har også mindre workshops/BOFs (om aftenen) om følgende emner (flere kommer løbende til, også under selve lejren)
* TCP-IP netværk for begyndere Kristen Nielsen
* Backup og Backup løsninger: (flere oplægsholdere)

Nyhedsbrev - juni

Herudover er der naturligvis tid til hygge både foran computeren, ved lejrbålet, når vi laver og spiser mad osv.

Lejren arrangeres af ølejrbevægelsen, DKUUG yder vigtig støtte for at lejren kan afholdes, bl.a. til etablering af netforbindelse mv.

Læs mere om The Camp på http://www.thecamp.dk

Kontaktperson : Kristen Nielsen <krn@dkuug.dk>

LinuxForum

Coord-holdet planlægger en bof-dag (diskussionsgrupper) hvor det igen bliver DKUUG, BSD-DK og SSLUG som står bag.
Det bliver sandsynligvis 6. november 2004.

Dagen bliver i LinuxForum-stil, men pt. er ideen kun at lægge bof-er ind, så der er tid til at snakke teknik, udveksle ideer, og hygge sig.

Et af de problemer der er mht afholdelse af LinuxForum er brandtilsyn i Symbion. Det er ikke klart hvor mange personer, der må være i Symbion.
Hanne Vilmann undersøger nærmere.

coord-holdet der planlægger LinuxForum-konferencen er blevet udviddet med Kenneth Geisshirt, som er et kendt ansigt i både DKUUG og SSLUG.

Kontakt : <coord@linuxforum.dk>

KLB - DKUUG klub-aktiviteter

KLB afholder DKUUG klubmøde den sidste tirsdag i hver måned i M4 i Symbion. Annoncering sker på denne postliste, på <announce@opensource.dk> og på http://www.dkuug.dk.

Den 25. maj afholdt DKUUG klubmøde om Linux-klynger med Kenneth Geisshirt som taler. Kenneth holdt et veloplagt foredrag om forskellige måder at udnytte sammenkoblede Linux-maskiner, enten til udregning af kæmpeopgaver, eller med henblik på at sikre høj oppetid (fail-over), eller fordeling af belastning (f.eks. webserver). Fokus lå på open source løsninger, men flere kommercielle og lukkede løsninger blev også omtalt.
Kenneth forklarede også en del om de problemer, der ligger i at designe klynge-systemer, f.eks. i form at begrænsninger i NFS, eller låsning af filer mellem maskiner der evt. alle skal bruge samme fil. Slides kan findes på http://kenneth.geisshirt.dk/talks/klynger.pdf

Næste klubmøde er tirsdag den 29/6

Kontaktperson : Sidsel Jensen <sj@dkuug.dk>

Nyhedsbrev - juni

NET - DKUUGs netværk og drift-udvalg

I maj holdt NET sin anden arbejdsweekend, hvor vi arbejdede med omlægningen af STD-serveren (std.dkuug.dk) fra gammel hardware og styresystem (Red Hat 5.2) til ny hardware og en tidssvarende FreeBSD 5.2.1. Alt indhold blev flyttet og verificeret at det er helt korrekt (ved at tage md5 checksummer på alle filer på maskinen).
Herudover er anvendt en væsentlig mængde timer på support af STD på både teknisk og indholdsmæssigt niveau. Der blev fundet et par fejl.
De fleste fejl kunne ved nærmere analyse også findes i den gamle opsætning.

Der er blevet arbejdet videre med flytningen og opdateringen af indhold på den nye ftp server. pt. er apache-mirror startet op på ny server. (apache-mirror.dkuug.dk)

Flytning af indhold fra den gamle webserver er foretaget, herunder: betalingssystemet (som dog skal testes endeligt inden dette er klart), nordu-siderne (nordu.dkuug.dk), og mange andre småting, som skal køre videre på den nye webserver.

Vi har så småt taget fat på drøftelserne af krav og teknik til et nyt backup system.

I juni forventer net at holde et møde, og ligeledes et sommermøde i juli .

Kontaktperson : Kristen Nielsen <krn@dkuug.dk>

STD - DKUUGs standardiseringsliste

Hjemmeside : http://std.dkuug.dk

Som omtalt i sidste måneds nyhedsbrev, så har DKUUG sammen med en række IT-organisationer arbejdet med definitionen af åbne standarder i lang tid. På et møde i starten af maj rettet STD de sidste detaljer til og lavede version 1.1 af definitionen af åbne standarder http://www.aaben-standard.dk/.

Efter kort tid har denne definition opnået støtte fra følgende organisationer
* DKUUG
* SSLUG
* PROSA
* IT-Pol
* OSL
* Internet Society (DK)
* BSD-DK
* Digital Forbruger Danmark
* KLID

Kender du/I andre som ønsker at støtte op, så læs mere på http://www.aaben-standard.dk/.
Peter Mogensen har desuden lavet en engelsk oversættelse af den danske tekst, som nu er at finde på http://www.openstandard.dk/.

Nyhedsbrev - juni

I maj har der været en hel del blæst omkring driften af std.dkuug.dk.
Maskinen blev i starten af maj omlagt til at køre FreeBSD efter at den har kørt Red Hat 5.2 i mange år. Dette medførte efterfølgende at Keld Simonsen flyttede en hel del af webstedet over til sit eget domæne http://www.open-std.org/.
Der har været en aktiv diskussion om dette var fornuftigt. Spørgsmålet bliver taget op af de respektive standardiseringsgrupper samt af DKUUGs bestyrelse.

Kontaktperson Peter Toft <pto@dkuug.dk>

BLD - redaktion af DKUUG-nyt

DKUUG-Nyt nr. 148 er lige på trapperne.

I denne udgave af bladet kan man bl.a. læse 2 udvalgte papers fra NordU-konferencen, samt en artikel om skak-verdensrekord-forsøget som DKUUG var vært for ved samme lejlighed.

BLD-udvalget er desuden gået i tænkeboks for at brainstorme over hvilke muligheder, der er for at etablere en alternativ kommunikationskanal fra DKUUG til medlemmerne. Hvis uheldet er ude og DKUUG ikke bliver accepteret i den nye bladpulje, som vi har ansøgt om at blive optaget i, vil portoudgiften til udsendelse af DKUUG-Nyt blive meget større end den er i dag. Dette bevirker enten, at der udgives færre DKUUG-Nyt i løbet af året, eller at vi f.eks. udsender nogle blade elektronisk.

Der vil i den kommende tid sandsynligvis blive oprettet en postliste, som man kan tilmelde sig, hvis man ønsker at modtage DKUUG-Nyt som PDF-fil. Listens navn vil blive annonceret på DKUUGs hjemmeside, samt i et kommende nyhedsbrev.

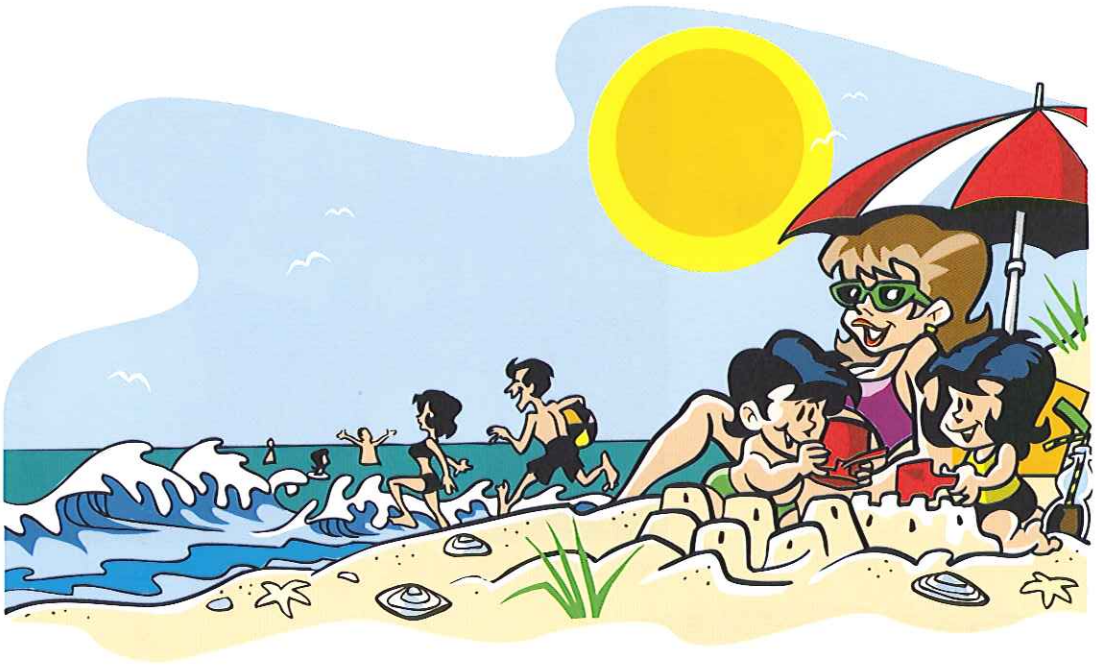
Kontaktperson Hanne Vilmann <sek@dkuug.dk>

Opfølgning på nyhedsbrev og deltagelse i udvalg

Har du lyst at være med i DKUUGs udvalg, så skriv til det enkelte udvalg at du har lyst til at deltage eller tag kontakt til den person der er nævnt ved det enkelte udvalg.

DKUUG har også en debat-liste <debat@dkuug.dk> som er til diskussion
om foreningen og gerne til opfølgning på dette nyhedsbrev.

Tilmelding til debat-listen og denne aktivitets-postliste sker via
http://www.dkuug.dk/content/view/87/26/



Sekretariatet holder lukket alle fredage i juni, samt alle dage fra den 19. juli til den 6. august 2004.

DKUUG ønsker alle en god sommer

Næste nummer af DKUUG-Nyt

Hvorfor ser de sådan ud ?

Foto: Peter Toft

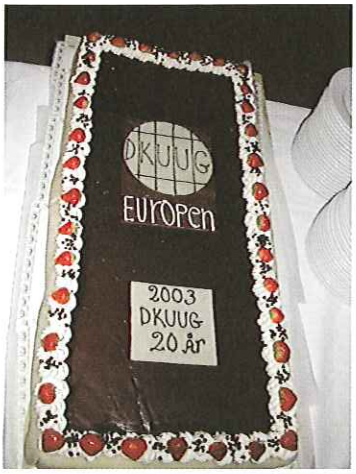
Kommer hun mon op og flyve ?

Foto: Kristian Vilmann

Hvormange mennesker kan man presse ind til LinuxForum ?

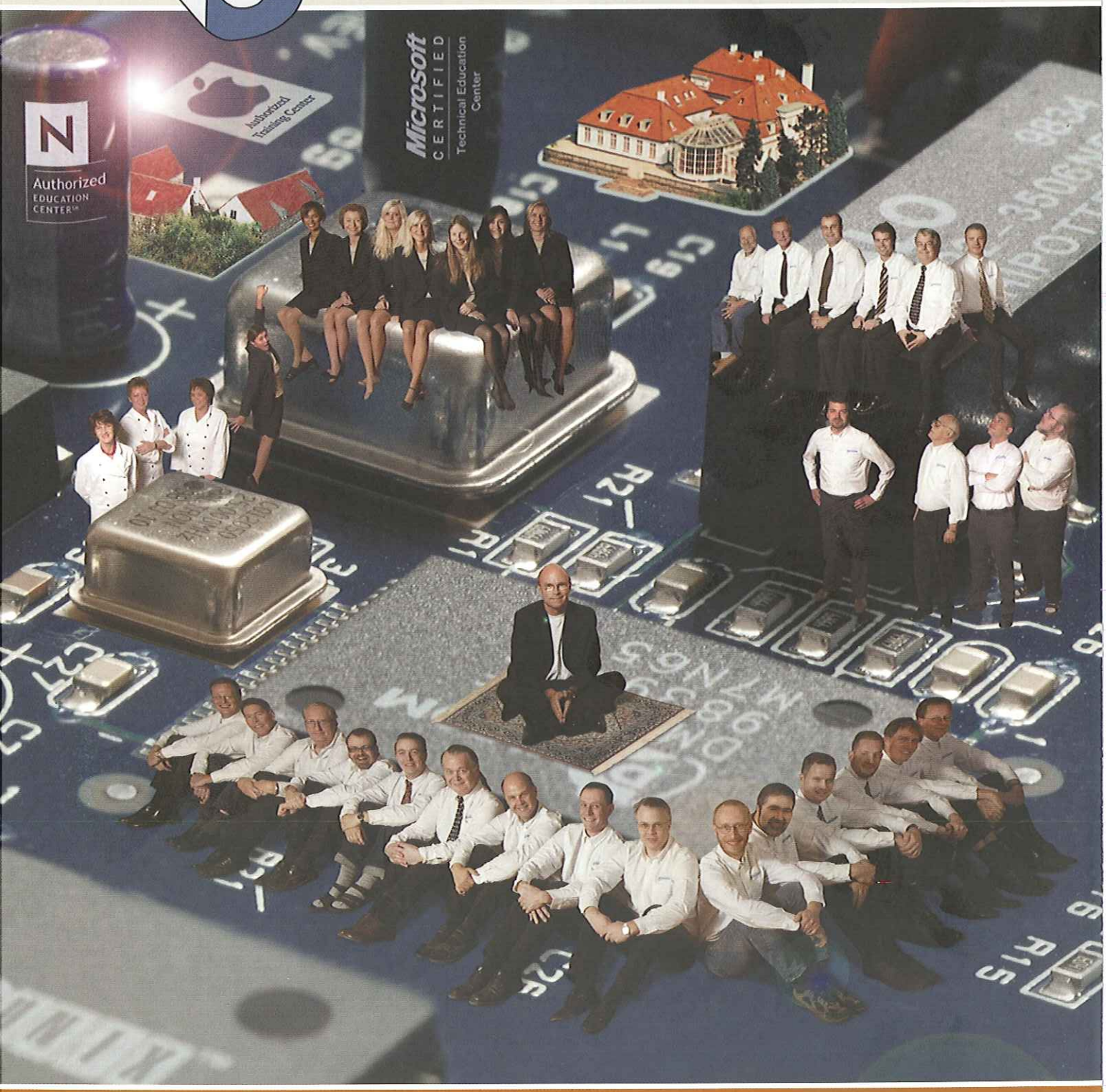
Foto: Hans Schou

Billeder fra DKUUGs jubilæumsfest



Bestil nyt 368-siders 2004 gratis kursuskatalog
katalog@superusers.dk eller Tlf. 48 28 07 06

SUPERUSERS 2004



Danmarks største it-kursuskatalog

Kurser • Konsulenter • Certificering
C/C++, C#, Java, Perl, VB.NET., SQL
TCP/IP, VoIP, VPN, XML
UNIX, Windows, NetWare