Doc. Identifier:
DataGrid-04-NOT-xxxx

Date: **27/03/2002**

GSI and encryption in the Fabric

Subject:	**Connection speeds for GSI and plain TCP connections and the speed of encryption**
Author:	**David Groep**
Partner:	**NIKHEF**
Diffusion:	**unlimited**
Information:	

1. INTRODUCTION

For connections within the fabric authentication and authorization services will be required that are sufficiently fast to support monitoring data transports for O(10^5) nodes. But we have to keep in mind that not all compute fabrics will operate on a trusted network: there will be sites that have their compute power distributed over many desktop workstations. In such cases, eavesdropping and man-in-the-middle attacks have to be reckoned with. Therefore, configuration and monitoring information sent to fabric elements must be confidentiality and integrity protected. The approach chosen should prevent eavesdropping, man-in-the-middle attacks, and replay-attacks.

These requirements are usually satisfied by asymmetric cryptography applied to communications sessions, partly by conventional TLS and more in full by GSI.

This note was triggered by the following mail by Sylvain Chapeland from March 26, 2002:

```
In particular, we will have a mechanism to subscribe to metrics on the
repositories (local and central). Of course we want to control access to
these services (and if possible with simple configuration ...). The idea
is that once a client has registered one or several metrics, it has a
listening port on which it receives newest data when available. It is
not possible to keep all connections open all the time because there
might be a lot of them. Therefore the authentification should be not too
heavy to handle each time we re-connect.

We also need such way to authenticate client/servers in the transport of
monitoring data from local nodes to central repository.
```

Based on the discussion below, I would propose to apply full GSI authentication to the subscription request, followed by symmetric block ciphering of the subsequent data streams.

The same mechanism can be applied to the node–server communication, if this involves an initial "registration" connection. If no such connection is to be established, a permanent host-based key can be used and stored in the CDB. This does require secure installation services

2. MECHANISM COMPARISON

2.1. FULL GSI WITHIN THE FABRIC

The WP4 internal security architecture defined a Fabric-local ID service (FLIdS) component [1], that can act as an automated, policy based CA for intra-fabric authentication. This will allow the use of GSI protected connecting between machines, between services and between operators and services. For this to work, all the principals involved must have a unique credential (certificate), so both the servers and every possible client. This looks like a lot of work (and probably is), but remember that the operators that start the monitoring client will probably have a certificate already, and that the automated systems will avail over such a key since that will also be acting as servers. The access control lists that should be used by clients can use the names issued by the FLIdS (hostnames, IP numbers or similar).

In this scheme, the partners are authenticated on every connect. Given the size of the certificates, this may imply an overhead of several kilobytes. In a typical exchange, some 14 messages are sent, varying in size between ~50 to ~2000 bytes.

Tests show that full authentication on every connect does not scale well to a large number of connections per unit time: every connection takes about 28 ms to set up. This 28 ms is only due to the TLS/GSI authentication overhead, since no further processing was done in this test. Full authentication is therefore not a viable option.

2.2. INITIAL GSI AUTHENTICATION ONLY

In the spirit of GSI and TLS, one could use asymmetric keys and full GSI authentication only when requesting a subscription. As this time, a symmetric key could be exchanged securely, and this key is used to encrypt any data sent to the listening port on the client (subscriber) side.

This further message exchange can then use a conventional block cipher. Since only the two parties involved avail over the (symmetric) key needed to read the data, you do not need any further authentication. In fact, it is just extending the SSL session over multiple connections.

The connection overhead is almost zero. The typical time needed to set up a TCP connection is approximately 70 μs, about 380 times faster than a GSI authenticated connection.

The crucial factor then is encryption speed. Since you need to establish the encryption context only once, the overhead involved in context generation can be neglected. On a typical system (PIII, 1.2GHz, memory-to-memory), the encryption speed is 37 sec/GByte, i.e., 34 ns/byte. The conventional block length is 8 bytes for the most popular block ciphers (CAST, IDEA, Blowfish, etc.)

Both the client and the information provider do have to maintain some encryption state about the subscription. Alongside with the port number to connect to with new data, the encryption context (cipher, key) must be stored. Also the client must hold this key to be able to use the data sent to it.

2.3. OTHER OPTIONS

No other options are considered in this note.

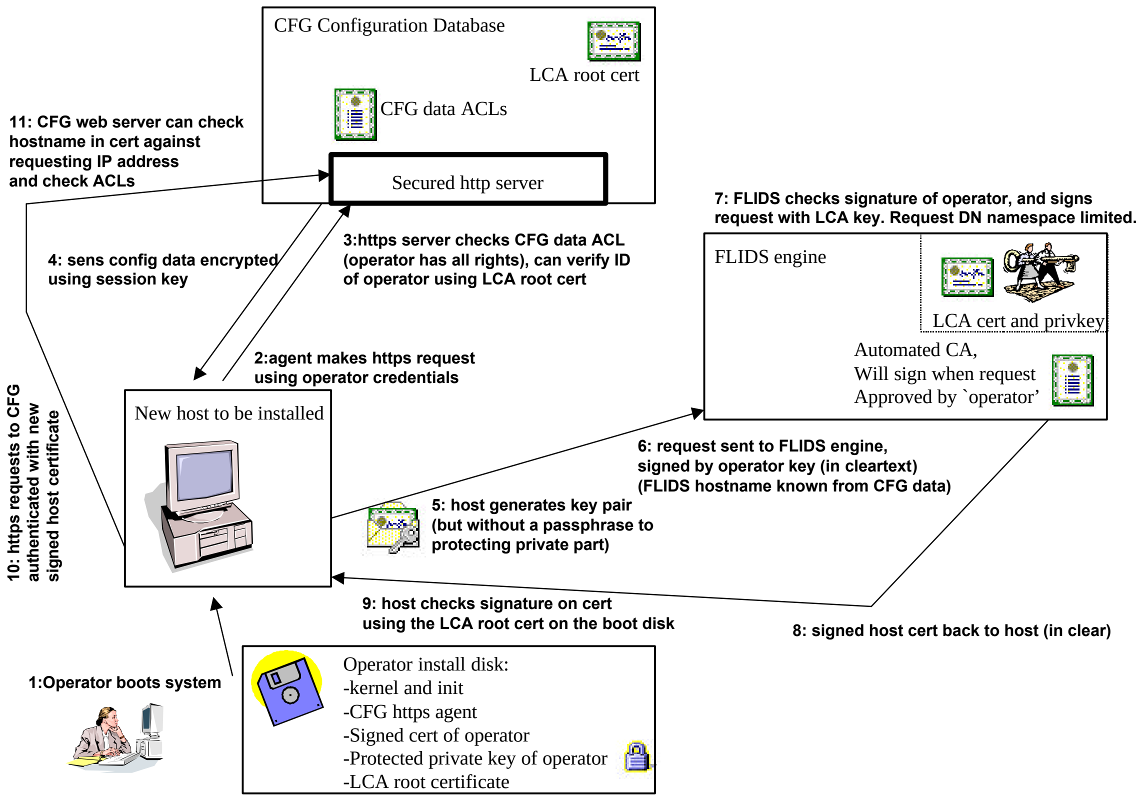
2.4. GETTING THE HOST KEYS TO THE SERVERS

Distributing host and service keys to newly installed systems was described in the FLIdS use case. The relevant flow diagram is reproduced in Appendix A for informational purposes only.

2.5. LIMITATIONS

This approach will only suffice for message *content* confidentiality and does not protect against traffic analysis.

3. FABRIC LOCAL ID SERVICE USE CASE


CFG Configuration Database
LCA root cert
CFG data ACLs
Secured http server
11: CFG web server can check hostname in cert against requesting IP address and check ACLs
7: FLIDS checks signature of operator, and signs request with LCA key. Request DN namespace limited.
4: sens config data encrypted using session key
3:https server checks CFG data ACL (operator has all rights), can verify ID of operator using LCA root cert
FLIDS engine
LCA cert and privkey
Automated CA, Will sign when request Approved by `operator'
2:agent makes https request using operator credentials
10: https requests to CFG authenticated with new signed host certificate
New host to be installed
6: request sent to FLIDS engine, signed by operator key (in cleartext) (FLIDS hostname known from CFG data)
5: host generates key pair (but without a passphrase to protecting private part)
9: host checks signature on cert using the LCA root cert on the boot disk
8: signed host cert back to host (in clear)
1:Operator boots system
Operator install disk:
-kernel and init
-CFG https agent
-Signed cert of operator
-Protected private key of operator
-LCA root certificate

4. REFERENCES

[1] **D4.2** Archtectural Design and Evaluation Criteria WP4 – Fabric Management, page 24.

5. TABULAR DATA

All tests were performed on a PIII 1.2 GHz IBM e-Server xSeries 330 duel-processor system over the loop back interface. All tests were memory-to-memory when relevant.

5.1. PLAIN TCP CONNECTIONS

#connections made	wallclocktime [ms]
0	2
1	3
2	3
100	9
1000	74
10000	687

The average time per connection in the infinite limit is ~ 70µs.

5.2. GSI CONNECTIONS

The GSI connections were self-secured, with message integrity and confidentiality enabled. The code was based on the sample implementations from Globus_io (connect and listen).

```
time ./connect -p 56806 -h localhost -a \
      -u self -m gssapi -d safe -d private -I conncount
```

#connections made	wallclocktime [ms]
100	2675
1000	27056

The average time per connection in the infinite limit is ~ 27056 µs.

5.3. BULK CRYPTOGRAPHIC CAST TEST

A sparse-file of zeroes was encrypted using the CAST algorithm and the results were written to /dev/null.

#bytes encrypted	wallclocktime [ms]
1.07 M	53
10.4 M	385
104 M	3722
1.0 G	37171

In the infinite limit, CAST will encrypt one block (8 bytes) every 272 ns on the test system.