ProFocus: A Fault-Tolerant Key Storage Service with Proactive Recovery

Punit Agrawal, Carl Hauser

Goals

- Develop a service to store and distribute signing and encryption keys needed for secure, authenticated delivery of messages originating at GridStat publishers and sent to multiple subscribers
- Ensure that the system can tolerate a pre-specified number of *disclosure failures* and enable the service to remain available during a pre-specified number of *Byzantine failures*
- Ensure that none of the nodes comprising the service ever learn the stored secrets

Fundamental Questions/Challenges

- Key storage and dissemination services can be a weak point in overall system security.
  - Even just one compromised node can disclose shared secret keys and highly sensitive information.
  - Unavailability of the service can cause communication issues.
- No node in systems that replicate and distribute data across multiple nodes can be trusted.
- Asynchronous distributed systems can experience an unbounded number of failures but can tolerate only a bounded number of them.

Research Plan

- Use threshold cryptography to withstand compromise of storage service nodes and to enhance availability
- Use an *opaque masking Byzantine quorum system* to implement replicated data storage.
  - $5t$ servers for t- fault tolerance.
  - At most t of n servers compromised.

Type of Quorum system	Number of servers n	Quorum Size $\|Q\|$
Opaque Masking	$n \geq 5t$	$\frac{\lceil 2n + 2t \rceil}{2}$
Dissemination	$n \geq 3t + 1$	$\frac{\lceil n + t + 1 \rceil}{2}$

Table 1: Byzantine Quorum Systems

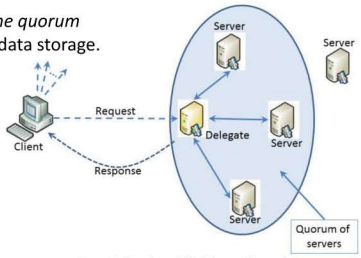
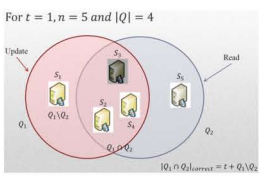


Figure 1: Overview of Client Request Processing

- Use *proactive recovery* to tolerate an unlimited number of failures over the life of the system even though only t faults can be tolerated within a rejuvenation interval.
- Use *public-key cryptography* to ensure confidentiality and integrity of stored secrets and authenticity of request-response messages.
  - **service key pair** $<sk_{pub}, sk_{priv}>$.



- Use $(n, t+1)$ *threshold cryptography* to distribute shares of the sk_{priv} among servers.
- Recovering a stored secret requires the collusion of $(t+1)$ servers.

- **ProFocus**:
  - Stores name-secret bindings of the form $< \tau, N, E_{sk_{pub}}(k) >$: timestamp, name, secret.
  - Basic opertions: read_secret, read_timestamp, read_masked_secret, update.
  - Reading a secret.
    - Client generates blinding factor b and sends $\mathrm{E}_{sk_{pub}}(b)$ to server.
    - Each server i computes partial decryption $D_{sk_{priv_i}}\left(E_{sk_{pub}}(k) * \mathrm{E}_{sk_{pub}}(b)\right)$.
      - Note: requires homomorphic encryption system.
    - Delegate server combines partial decryptions to get $(k * b)$ using threshold cryptography and sends back to client.
    - Client retrieves $k = (k * b)/b$.

Research Results

- ProFocus integrated with GridStat to improve pub-sub communication security.
  - Prevent key leakage by compromised leaf security management server.
  - Key generation responsibility moved from leaf-SMS to Publisher and stored as secrets in ProFocus.
  - Subscribers retrieve publication keys from ProFocus.

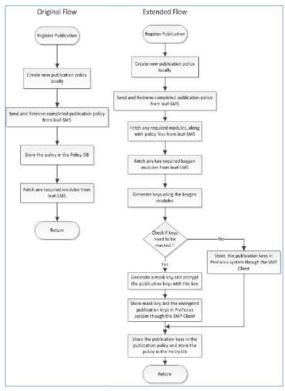


Figure 2: Registration Process of New Publications with ProFocus

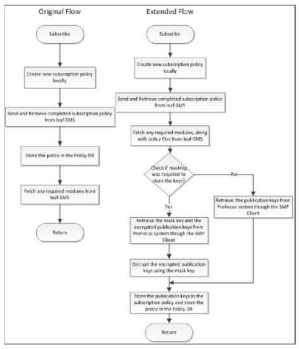


Figure 3: Registration Process of New Subscriptions with ProFocus

	64	128	256	512
Read_Secret	1334.80	1311.56	1339.39	1359.26
Read_TimeStamp	599.16	620.81	591.06	615.30
Read_Masked_Secret	620.28	633.40	616.54	626.08
Update	620.21	635.15	618.41	620.92

Figure 4: Performance of ProFocus Operations with 5 Servers and Fault-threshold 1

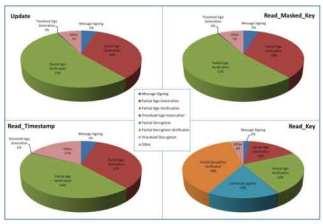


Figure 6: Cost of Individual Steps of Each ProFocus Operation

	64	128	256	512
Read_Secret	1232.76	1233.62	1246.45	1243.15
Read_TimeStamp	563.20	620.81	573.32	563.78
Read_Masked_Secret	590.20	633.40	679.98	628.28
Update	569.26	551.91	578.79	566.24

Figure 5: Performance of ProFocus Operations with 5 Servers and 1 Faulty Server

Broader Impact

- Secure storage of keys is needed for many purposes:
  - Wide-area communication.
  - AMI.
  - Field devices generally.
- No single business entity has unilateral access to others' secrets.

Future Efforts

- Design and implement access control mechanisms for secret storage and retrieval.
- Automate triggering of proactive recovery: Assess security properties of various triggering policies.
- Implement proofs of plain-text knowledge in client-service and server-server protocols.