because the service only knows that secret key:

$$nonce = time + serialnumber + MAC(time + serialnumber, secretkey)$$

So the nonce is not only cryptic but enables the service to read the time and serial number. Furthermore no synchronization of clocks is necessary, because the nonce is sent via a challenge-response technique. Hence the service compares the time bound to the nonce only with its own clock.

The service has two instruments to tighten the security. On the one hand it can limit the life time of a nonce by restricting the time interval in which the nonce must be responded by the client. The XKMS web service verifies that the nonce represented was generated recently. In addition the MAC value is checked. On the other hand a service is not only checking the time frame and the MAC, but in addition it tracks all used nonces. The service checks that the nonce was not already responded before. The amount of stored nonces is limited due to deleting those nonces which are expired.

To ensure eventually more security, the nonce is encrypted additionally. That operation masks the values of the serial number and time, but implies higher computing effort.

5.4 Security improvements

The following three found suggestions for improvement of the security of XKMS matter only in case of that a communication is not secured by transport layer security, but only by message layer security. Each attack is based on a replay of previous requests or responses. In fact the specification note in paragraph 355 of XKMS 2.0 [1] that "precise mechanism by which replay attacks are prevented is left to the implementation", but that does not consequently mean that an XKMS implementation must decide under which circumstances replay attacks must be provided. That is the task of the specification to mention those cases. The specification does not create awareness of the necessity of replay attack prevention in the below described cases.

5.4.1 Random generation of the message IDs

Granted that each message is protected with message layer security and that message correlation is done by the *RequestId* attribute. Further on Carol has the possibility to query in the same manner as Bob the X-KISS Tier II service. She knows from previous queries of Bob the structure and systematic of his message IDs. So she can foresee which IDs Bob will use in future.

Under these circumstances Carol queries the service with those IDs to get an assertion about the trustworthiness and validity of Alice's public-key binding and stores the responses. Thereupon she steals Alice's private-key and does not care about that Alice subsequently revokes her public-key bindings. Later on Bob wants to encrypt confidential data with Alice's private-key. Therefore he queries the X-KISS Tier II service, to get an assertion that her key-binding is still valid and trustworthy. Carol intercepts Bob's request and searches for her previously asked service's responses with the same *ResponseId*. She replays that Validate response to Bob, who still believes in the validity of the certificate.

That scenario is in case of X-KISS Validate and X-KRSS responses a dangerous issue, because the service asserts the trustworthiness of the returned public-key bound information. It is advisable that the specification makes a client aware how important it is to generate an unpredictable ID. It must explicitly oblige each client to make it difficult for Carol to guess IDs. The specification mentions shortly in paragraph 45 of XKMS 2.0[1] the unique and random generation. But a further explanation to make implementers conscious about the security effects of guessable IDs is missing. So it is advisable to alert an XKMS client implementer to the issue at least one time more, preferably in the section 2.5 of XKMS 2.0 Bindings[5].

5.4.2 Replay of X-KRSS requests

Not mentioned in the specification is the potential to do damage, if a client has executed two X-KRSS Reissue requests considering the same registered public-key binding and Carol replays the first X-KRSS Reissue request to restore the old state. Figure 5.1 on the next page depicts that attack.

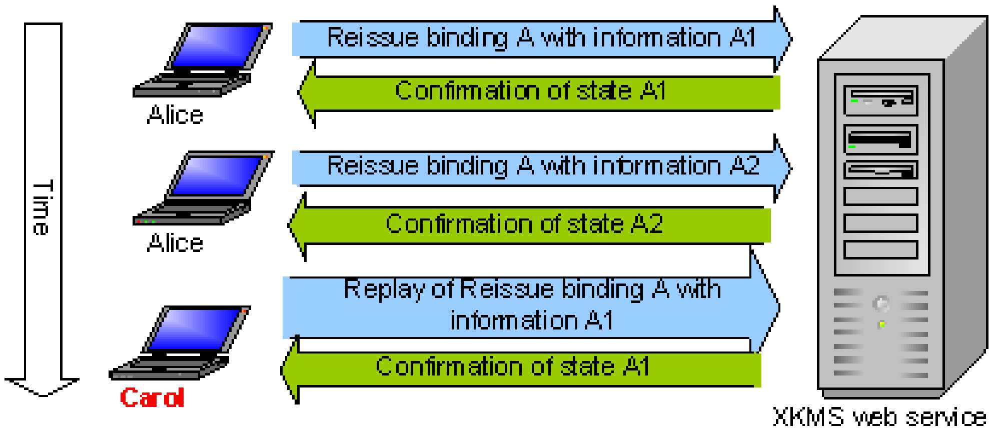
Time
Alice
Reissue binding A with information A1
Confirmation of state A1
Alice
Reissue binding A with information A2
Confirmation of state A2
Carol
Replay of Reissue binding A with information A1
Confirmation of state A1
XKMS web service

Figure 5.1: Replay attack of a Reissue request

That is only possible if in addition the policy of an X-KRSS web service, which is defined by the implementation as stated in paragraph 273 of XKMS 2.0 [1], allows several Reissue requests with the same previously out-of-band exchanged secret for authentication and authorization of requests. Hence the XKMS specification must suggest in paragraph 38 in XKMS 2.0 Bindings [5] that a replay of an X-KRSS Reissue request is a danger, if the authentication policy of an XKMS web service permits multiple X-KRSS Reissue requests with the same secret. In that case to make that replay attack more difficult, it is advisable to combine either the <*Authentication*> signature of a Reissue request with transport level security and/or with the <*MessageAbstractType*> signature combined with the Two Phase Request protocol, including a nonce against request replay attacks.

Similar considerations are valid for a replay of Alice's registration request, after Alice previously revoked her public-key binding. But the probability that the policy of an XKMS implementation accepts in the second Register request the same authentication and authorization code as in the first Register request is lower than accepting the same secret for several Reissue requests, because the replay attack of Register requests is more obvious. Nevertheless it is advisable to note also that attack in the specification.

5.4.3 Replay of an X-KISS Tier 1 response

A replay of an old CRL in an X-KISS Locate Response is a difficult but possible attack. The XKMS 2.0 Bindings specification [5] specifies in paragraph 38, that response authentication is not mandatory in case of an X-KISS Locate Service, if self authenticating data like a certificate and/or Certificate Revocation List is sent.

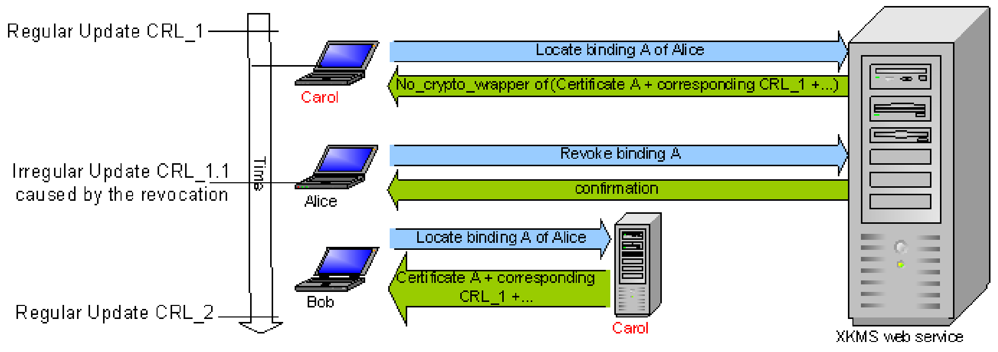
Regular Update CRL_1
Irregular Update CRL_1.1 caused by the revocation
Regular Update CRL_2
Time
Carol
Alice
Bob
Locate binding A of Alice
No_crypto_wrapper of (Certificate A + corresponding CRL_1 +...)
Revoke binding A
confirmation
Locate binding A of Alice
Certificate A + corresponding CRL_1 +...
Carol
XKMS web service

Figure 5.2: Replay attack of a not fresh CRL

Figure 5.2 depicts a replay of an old CRL. Supposed that Carol is in possession of a previous Locate response, even possible queried by her, a replay of that message is not prevented because already authentication of the response is not recommended by the specification. If Bob then queries the X-KISS Locate service to retrieve the self authenticating public-key information and the corresponding CRL, he can be faked by Carol, who replays an older version of the CRL. That CRL_1 does not contain Alice's already revoked certificate listed in the freshest CRL_1.1. The pre-condition is that the policy of the XKMS web service permits an irregular update of the CRL_1 with the CRL_1.1.

But also in case that it is excluded in an XKMS web service's policy to generate intermediary CRLs, care must be taken. Although already CRL_2 exists, Carol replays CRL_1. It can be argued that Bob recognizes the old CRL by means of its generation time stamp and update time, but that requires synchronized clocks between Bob and the issuer of the CRL. That requirement of synchronized clocks is a difficult task, because XKMS tries to enable flat clients like mobile phones to use XKMS.

Thus in both cases a client queries an XKMS Tier 1 Locate service and cannot be sure that the information returned is the freshest one. So it is advisable to change the XKMS specification and obligate each XKMS implementation to ensure data integrity and authentication of at least services' responses. Hence Carol will have more difficulties to fake a Locate response, because Bob's request is correlated to the Locate response with the *ResponseId* attribute.

Any discrepancies in XKMS 2.0

In the following proposals for improvements of the XKMS 2.0 Working Draft [1] are made:

- In example 2.5.3.3 the *RequestId* and *ResultMajor* are missing in the notification message. Clients has no means to differentiate the corresponding response from other responses of other asynchronous processed requests.
- In paragraph 85, the scope of the <*ds:Signature*> is not only the entire ”request” message, but it holds for each message. Nonce not only used against replay attacks in combination with message level security, but also to make denial-of-service attacks more difficult.
- In paragraph 96 and in paragraph 113: append string ”and elements” to the string ” ...contains the following attributes”.
- Paragraph 126, move element definition to paragraph 125.
- Response examples, e.g. paragraph 147, <*KeyUsage*> contains data from the type *QName*, so that the prefix ”xkms:” is needed.
- Paragraph 148, <*KeyName*> requested in text, but not in the corresponding listing of paragraph 149.
- Paragraph 241, <*RevocationCodeIdentifier*> element is missing and does not suit to the Revoke request example in paragraph 259.
- Paragraph 254, a client cannot change the public-key bound information by providing a new certificate, because only the service creates certificates. Hence new data is missing to change the binding, the purpose of the Reissue request.
- Paragraph 273, misleading name, since dubious if ”Registration services” is the same as ”X-KRSS services” as supposed within this diploma thesis, or is the statement only valid for Register messages? Similar problem exists in paragraph 289 with ”registration request”.

- Table in paragraph 351 under Security Enhancements, no security required for Locate operations should be reconsidered due to the hints of security improvements in section 5.4.3
- Paragraph 365, The <*PassPhraseAuth*> element is already excluded in the related XKMS 2.0 Schema. Probably leftover of the XKMS Note [2].
- Paragraph 38 of the XKMS 2.0 Bindings [5], The statement "A service only needs to *authenticate* a requestĚif either the information is *confidential*Ě" does not suit, because encryption must be applied and not authentication in case of confidential information.
- Appendix C.1.2 and C.1.3,Authentication Data does not suit to the previously provided examples.
- Appendix C.1.4, Computation steps incomplete.
- Appendix C.2.1 Used keys does not suit. Change "0x1" to "0x2" and "0x2" to "0x3"