Corporate Signatures Mediator Server Guidelines

Aleksander Lasecki, 236371
Grzegorz Zaborowski, 236447
Paulina Jałosińska, 232892
Mikołaj Grzegrzółka, 241073

Contents

1 Introduction

The following document is a set of guidelines for implementing corporate signature mediator servers which aim to conform with the Protection Profile "Corporate Signatures Mediator Server Protection Profile" written by the same authors. It focuses on the main security mechanisms of such an implementation. A brief mapping of them to cryptographic techniques is given in **Figure 1**. Those techniques are introduced later in this chapter and described in greater detail in the following chapters. **Chapter 2** given a detailed description of how each individual procedure of the mediator system is supposed two work. **Chapter 3** add recommendations of specific key lengths and algorithm to be used in implementations of those procedures and **Chapter 4** gives a listing of ANS.1 specification of all necessary data structures.

Mechanism	**Protection**	**Technique**
Confidential file signing during specified hours	Authenticity and Confidentiality	Parametric digital signature
Key generation	Confidentiality	Partial private key infrastructure
Setting signature parameters	Authenticity	Secure authentication
Accessing revocation database	Authenticity and Confidentiality	Secure authentication

Figure 1: A brief listing of the main security mechanisms

1.1 Confidential file signing during specified hours

Corporate signature generation procedures implemented on the server need to be designed with the following features in mind:

- An employee's ability to create signatures should be easily revocable by the company
- Both the employee and the company should be able to determine specific time periods when signatures can be successfully created (e.g. only on work days and not during time off)

Because of that, those procedures require a private key solution that ensures the participation of both parties (i.e. the mediator server and the employee that issues the signing request). To achieve this, the system uses a partial private key setup in which the private key is broken down into two parts, one is stored within the mediator's secure storage and the other on a device owned and controlled by the employee. For a signature to be successfully created and verifiable with the combined public key, both parties are required to participate in the protocol and perform computation with their part of the secret key.

1.2 Key generation

The two-party key generation protocol must be performed in a distributed manner. The design need to follow features as:

- No 'trusted third party' participation.
- Neither side knows the secret (half of the private key) of the other.
- Both parties know the public key.
- Prevent from (maliciously) choosing 'bad' group parameters.

Keeping the above guidelines is very important to be able to talk about secure corporate signatures. By eliminating the 'trusted third party' and carrying out the protocol so that the other party does not know the secret of the other (half of the private key), we ensure that it will not be possible to sign any message without two parties involved (even a malicious administrator with access to the entire server infrastructure).

In the next section we present a protocol for two parties to generate Schnorr key in a distributed manner. It is just an ordinary Schnorr public key, $\langle p, q, g, y \rangle$. However, the corresponding Schnorr private key, x, is split between two key halves, x_c and x_s, such that $x \equiv x_c + x_s \mod q$. A public key can be centrally generated along with two halves for the corresponding private key. Schnorr additionally relies on a cryptographic hash function, G, mapping arbitrary strings to elements of Z_q^*. We will assume G has been specified as a parameter of the scheme.

1.3 Signature parameters

The website for user authentication included in the guidline was created in the context of providing additional user security. This includes protection against the use of unauthorized user accounts or standard phishing. In addition, the website should have a reduced impact of potential data breaches. The overall design of the website was presented in the context of maintaining higher security standards.

2 Procedures

This chapter describes the procedures that are required to achieve the functionalities required by the security mechanism described in the previous chapter.

2.1 Authentication

To provide secure authentication all parties involved in a procedure should communicate over TLS and perform Mutual TLS Authentication Technical requirements for the interface: TLS version 1.2 which is to enable client authentication through a certificate.

2.2 Webpage for User's authentication

2.2.1 Website requirements

Requirements for the mediator's website: the web browser must be able to authenticate at TLS level, which will be the level of authentication (to use the same certificate as for finalizing signatures).
The website security scope should be supplemented with additional functionalities enabling basic control functions. We also assume here the possibility of introducing additional functionalities in order to enforce the basic methods of the website operation.

2.2.2 Two-factor authentication(2FA)

The next level of authentication is the user's password (2FA). To log in to the website user needs to have login and password. Requirements of the password

- At least 10 characters,
- A mixture of both uppercase and lowercase letters,
- A mixture of letters and numbers,
- Inclusion of at least one special character, e.g., ! @ ?] (Note: do not use <or >in your password, as both can cause problems in Web browsers.)

. In the next step the user will be required to use software-based token:

- the application for the token should be installed on business smartphone,
- the smartphone should be also properly secured by the code or a fingerprint.

The users can use the standard soft token procedure. Each token will contain basic information about the user. The entire procedure will be generated by the administrator and assigned to each user. So each token can be defined as a separate piece of software. Each token will be secured with a PIN code stored directly on the remote server for authentication. If user will try to guess it several times, the token will be blocked by the authentication server.

2.2.3 Website security implementations

Additional support for the general operation of the website and ensuring an appropriate security standard for the user. We distinguish the following sub-points to improve the overall security of the website's functionality.

- Isolated implementation of web service components
- Additional security for website services related to the internet

- Separate interfaces for administration
- A secure process related to the users account recovery
- Additional layers for authentication during website activity
- Secure procedures for each operation
- Implementation of scanning tools for website vulnerabilities

2.2.4 Website options

Access to the user's account on the website has only the owner of the account. In case of forgotten password administrator has a right to reset the password. The website interface includes options such as:

- the user has the option to select the time periods when the user has the right to use the electronic signatures,
- the website will allow the user to check the signing history (provides logs from the signature finalizations, the user can check when the signature was used).
- additional setting like possibility if changing the password
- possibility to set time limits when authentication of a given user will not be possible
- the possibility of automatic exclusion from the possibility of authorizing a specific user (employee dismissal, security supplement)

2.2.5 Revoking the key

Revoking the key will be necessary in situations:

- There is a suspicion that the key is compromised(if the risk is low, the admin have a right to only temporarily block the signing options). In case of revocation If the user is authorized to receive a new signature, admin can generate fresh key.
- User has lost the right to use the key (e.g. the user stopped be an employee of a given company).
- By order of the supervisor (Admin should have access to the list of authorized persons).

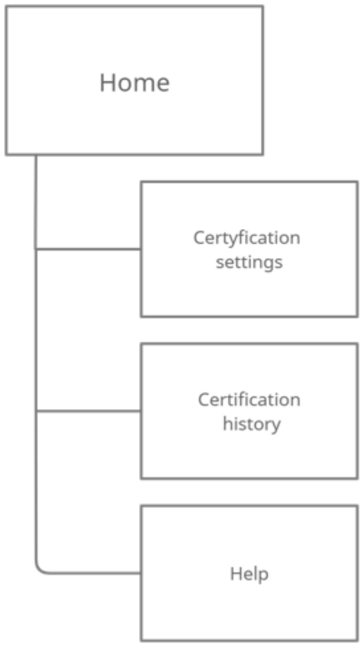

Figure 2: Scheme od the mediatr's website

2.3 Key Generation

A public key is just an ordinary Schnorr public key, $\langle p, q, g, y_i \rangle$. The corresponding private key x is split between the employee and the mediator server, such that $x \equiv x_e + x_m \mod q$. The half x_e corresponds to an employee, the half x_m corresponds to the mediator. Schnorr additionally relies on a cryptographic hash function, G, mapping arbitrary strings to elements of $\mathbb{Z}_q^*$. A public key can be centrally generated along with two halves for the corresponding private key as follows:

1. First the employee authenticates himself/herself to the mediator through TLS authentication and vice versa (described above).
2. Next both parties communicate in order to execute the key generation protocol. The employee chooses p, q, g (p is a large prime, q is a large prime such that $q|(p-1)$), and $x_e \leftarrow_R \mathbb{Z}_q$, computes $y_e = g^{x_e} \mod p$, and sends the server $\{p, q, g, G(y_e)\}$, where G is hash function.
3. The mediator picks x_m at random from $\mathbb{Z}_q$, computes $y_m = g^{x_m} \mod p$ and sends it to the employee.
4. Finally, the employee reveals y_e by sending it to the mediator.
5. Both parties compute $y = y_e y_m \mod p$, and the public key is $\langle p, q, g, y \rangle$

A shorter graphical description of the main protocol is given in **Figure 3**.

It is obligatory to perform additional checks to ensure that we are not dealing with malicious party. Before the step 3, the mediator should check if generated values p, q, g was generated according to some specific algorithm and according to the security parameters specification (we use the method proposed by the **NIST** in [5] or newer versions). Before the last step, the mediator should check that the received y_e corresponds to the previously received $G(y_e)$.

Verification of parameters p, q, g

To perform verification of parameters p, q, g following steps should be done:

1. Check if p, q fulfill the criteria specified in section 3.1.2.1.
2. Do a primality test for p and q.
3. Check range and order of g

$$2 \leq g \leq (p-1)$$
$$g^q = 1 \mod p$$

Failure of any of these tests is synonymous with an attempt to manipulate parameters. Such a situation should be recorded in the logs each time and an e-mail notification should be sent to the administrator responsible for this module.

$$\begin{array}{ccc}
\textbf{Employee} & & \textbf{Mediator} \\
q \leftarrow \text{a large prime} & & \\
p \leftarrow \text{a large prime such that } q|(p-1) & & \\
g \leftarrow \text{an element of } \mathbb{Z}_p^* \text{ of order } q & & \\
x_e \leftarrow_R \mathbb{Z}_q & & \\
y_e = g^{x_e} \mod p & & \\
& \xrightarrow{p, q, g, G(y_e)} & \\
& & x_m \leftarrow_R \mathbb{Z}_q \\
& & y_m = g^{x_m} \mod p \\
& \xleftarrow{y_m} & \\
y = y_e y_m \mod p & & \\
& \xrightarrow{y_e} & \\
& & y = y_e y_m \mod p \\
\text{Pub: } \langle p, q, g, y \rangle & & \text{Pub: } \langle p, q, g, y \rangle \\
\text{Priv half: } x_e & & \text{Priv half: } x_m
\end{array}$$

Figure 3: The key generation scheme

Certificate generation

After the public key has been established, a certificate needs to be generated be the company's Certification Authority (CA). To continue with the distributed notion of this signature scheme, both parties are needed for the certificate to be generated. This procedure requires that generating standard signatures is available for both parties (i.e. both, the employee and the mediator need to have their owe private key for generating a standard signature such as a Schnorr signature, specific algorithms are given in **Section 3**). The procedure assumes that all three parties area available and authenticated to each other via TLS. It follows the steps given below:

1. Both parties sign the public key $\langle p, q, g, y \rangle$ with their respective private key for a standard signature and send a request containing those signatures and the public key values to the company's CA
2. After obtaining and successfully verifying the signatures of both requests (which also includes making sure that the public key values sent by both sides are the same), the CA issues a new certificate for the newly generated mediated signature scheme

3. The certificate is the sent to the employee and the employee ensures that the values within it are correct

A simpler graphical representation of this procedure is given in **Figure 4**.

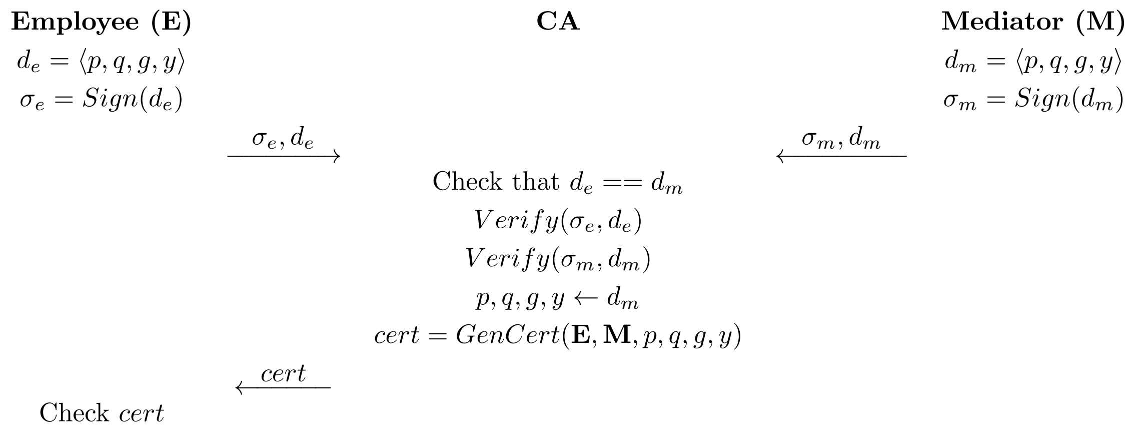

Figure 4: The certificate generation procedure

2.4 Signature Generation

The parametric signature generation procedure is based on a **Two-party Schnorr** signature scheme[2]. It uses the partial keys and other values generated in the **Key Generation** procedure, two hash functions G and H as well as the data base of signature parameters stored on a separate node of the mediator system to generate a signature in the following way (to simplify the description, the employee's device that takes part in the procedure is simply referred to as the employee. Also subscripts m and e are used to denote values computed by the mediator and the employee respectively):

1. First the employee who wishes to sign a message m authenticates himself/herself to the mediator through TLS authentication (described above) so both parties are authenticated to each other

2. Next both parties compute their secret ephemeral keys $k_m \leftarrow_R \mathbb{Z}_q$ and $k_e \leftarrow_R \mathbb{Z}_q$ and their corresponding public ephemeral keys $r_m = g^{k_m} \mod p$ and $r_e = g^{k_e} \mod p$

3. The main protocol is then initiated by the employee who sends the hash $G(r_e)$ to the employee as a commitment

4. The mediator receives the commitment, queries the revocation and signature parameters data base to make sure that the employee is allowed to generate a signature at the current time and if so sends back it's public ephemeral key r_m along with $G(r_e)$

5. The employee receives the response, makes sure that the received value of $G(r_e)$ matches the one computed locally and computes the randomness $r = r_e \cdot r_m \mod p$ and the inner part of a HMAC $h = H(K(r) \oplus ipad) \mid\mid m)$, where $K(r)$ is defined as $H(r)$ if r is larger the the block size or r otherwise and $ipad$ is a block sized padding made up of bytes with the value of $0x36$. It then sends r_m, r_e and h to the mediator

6. The mediator receives the response, computes the randomness $r = r_e \cdot r_m \mod p$ and the outer part of the HMAC $e = H((K(r) \oplus opad) \mid\mid h)$, where $opad$ is a block sized padding made up of bytes with the value of $0x5C$. It also computes its partial signature $s_m = k_m + x_m e \mod q$ and sends it to the employee

7. Finally the employee proceeds to compute the outer part of the HMAC $e = H((K(r) \oplus opad) \mid\mid h)$, their partial signature $s_e = k_e + x_e e \mod q$ and the full signature $s = s_e + s_m \mod q$. The employee then checks that the signature can be successfully validated and if so the employee outputs the final signature $\langle s, r \rangle$

$$\begin{array}{ccc}
\textbf{Employee} & & \textbf{Mediator} \\
k_e \leftarrow_R \mathbb{Z}_p & & k_m \leftarrow_R \mathbb{Z}_p \\
r_e = g^{k_e} \mod p & & r_m = g^{k_m} \mod p \\
 & \xrightarrow{G(r_e)} & \\
 & \xleftarrow{G(r_e), r_m} & \\
r = r_e \cdot r_m \mod p & & \\
h = H((K(r) \oplus \mathit{ipad}) \mid\mid m) & & \\
 & \xrightarrow{r_e, r_m, h} & \\
 & & r = r_e \cdot r_m \mod p \\
 & & e = H((K(r) \oplus \mathit{opad}) \mid\mid h) \\
 & & s_m = k_m + x_m e \mod q \\
 & \xleftarrow{s_m} & \\
e = H((K(r) \oplus \mathit{opad}) \mid\mid h) & & \\
s_e = k_e + x_e e \mod q & & \\
s = s_e + s_m \mod q & & \\
\text{Output } \langle s, r \rangle & &
\end{array}$$

Figure 5: The signature scheme

A shorter graphical description of the main protocol is given in **Figure 5**. The description above assumes that the employee performs their computation in accordance with the protocol. This assumption can be made since if one of the participants would provide incorrect values then the resulting signature would also be incorrect. The protocol uses a HMAC of the message instead of the message itself to protect its confidentiality and since HMAC is used instead of a simple hash, the value of h send over the communication channel will be different even if the same file is signed twice.

2.5 Storing private keys

After their creation, described in the **Key Generation** procedure, the partial private keys must be stored in a secure location, i.e. the mediators partial private key must be stored on it's HSM and the employee's partial private key must be stored on a TPM of the device designated to the employee by the company.

2.6 Storing signature parameters

The protection profile that these guidelines are referring to ensures that the mediator will be stored in a physically secure location. Because of that the signature parameters can simply be stored on a standard storage drive inside the mediator, however it is greatly recommended for this drive to be encrypted to increase the level of security.

3 Security parameter specifications

3.1 Signature security parameters

3.1.1 Authentication

Since authentication is done over TLS, it is required that **TLS 1.2** or later shall be available to all parties involved in the procedures of this system. Both the mediator and the employees need to be equipped with their own TLS Certificate. All certificates used for TLS authentication shall be compliant with the X.509 standard, version RFC5280[1].

3.1.2 Key parameters

3.1.2.1 Parameters specification: These parameters are defined as follows:

- p – a prime modulus, where $2L - 1 < p < 2L$, and L is the bit length of p. Values for L are provided below.
- q – a prime divisor of $(p - 1)$, where $2N - 1 < q < 2N$, and N is the bit legth of q. Values for N are provided below.
- g – a generator of a subgroup of order q in the multiplicative group $GF(p)$, such that $1 < g < p$.
- x_e, x_m – private key halves must remain secret and be randomly generated integers, such that $1 < x_e, x_m < q$.

Pair (L, N) shall be one of the following as recommended by **NIST**[7]:

- (2048, 224),
- (2048,256) or
- (3072, 256).

3.1.2.2 Random number generation: The algorithm used for generation of halves of private key (x_e and x_m) shall be one of the following as recommended by **NIST**[7]:

- Hash_DRBG
- HMAC_DRBG
- CTR_DRBG (with AES-128, AES-192 or AES-256)

3.1.2.3 Hash function: The hash function G used during the key generation protocol, according to **NIST** guidelines[7], the recommended hash function is any hash from the **SHA-2** or **SHA-3** family. Thus G shall be specified as one of the following:

- SHA-256
- SHA3-256
- SHA3-512

3.1.2.4 Prime number generation: In order to generate random prime numbers from a given range, it is recommended to use the following algorithms:

- **Miller–Rabin** test,
- Shawe-Taylor algorithm [6][8],
- others listed in NIST [6].

3.1.2.5 Primality test: To prevent an employee from (maliciously) choosing "bad" group parameters, the mediator should check if values p and q were generated according to some specific algorithm and specified safety parameters. To perform a primality test (or generating new primes), **Miller–Rabin** probabilistic primality algorithm [4][5] can be used. According to NIST [5] parameter $\mathbf{k}$ should be $\mathbf{k} \geq \mathbf{50}$, where k is the number of rounds performed.

3.1.3 Standard signature parameters

Standard (i.e. non-mediated) signatures are needed for the generation of a certificate. Signature generation should be available to both the mediator and the employee and signature verification should be available to the company's CA server. According to recomendations given by NIST[7], one of the following signature generation and verification algorithms should be used:

- **DSA** with parameters (L, N) with values $(2048, 224)$, $(2048, 256)$ or $(3072, 256)$
- **ECDSA** or **EdDSA** with $len(n) \geq 224$
- **RSA** with $len(n) \geq 2048$

3.1.4 Signature parameters

3.1.4.1 Hash functions: Two hash functions are used during the generation of a signature: G and H. According to **NIST** guidelines[7] the recommended hash functions are any hash from the **SHA-2** or **SHA-3** family. Thus both G and H shall be specified as one of the following:

- SHA-256
- SHA3-256
- SHA3-512

3.1.4.2 Random number generation: The algorithm used for generation of secret ephemeral keys at the start of signature generation shall be one of the following as recommended by **NIST**[7]:

- Hash_DRBG
- HMAC_DRBG
- CTR_DRBG (with AES-128, AES-192 or AES-256)

4 ASN.1 specifications

4.1 Key Generation

The key generation procedure described in **Chapter 2** uses three messages that are sent between the employee and the mediator. Those messages will be referred to as `Init`, `MediatorResponse` and `EmployeeResponse` respectively. Below we provide the ASN.1 type specification for all three messages of the key generation procedure along with the type of the resulting private key and public key:

```
Key-Gen-Schema DEFINITIONS AUTOMATIC TAGS ::=
BEGIN
    Init ::= SEQUENCE
    {
        primeModulus            INTEGER,
        primeDivisor            INTEGER,
        generator               INTEGER,
        publicKeyEmpoyeeHash    OCTET STRING
    }

    MediatorResponse ::= SEQUENCE
    {
        publicKeyMediator  INTEGER
    }

    EmployeeResponse ::= SEQUENCE
    {
        publicKeyEmployeee  INTEGER
    }

    PublicKey ::= SEQUENCE
    {
        primeModulus    INTEGER,
        primeDivisor    INTEGER,
        generator       INTEGER,
        publicKey       INTEGER
    }

    PrivateKeyHalf ::= SEQUENCE
    {
        privateKeyHalf   INTEGER
    }
END
```

Certificate generation

The certificate generation procedure described in **Chapter 2** is executed after the generation procedure. It uses two messages that are sent between the employee and the CA, and one message that is sent between the mediator and CA. Those messages will be referred to as `EmployeePart`, `MediatorPart` and `CertificateResponse` respectively. Below we provide the ASN.1 type specification for all three messages of the certificate generation along with the type of the resulting certificate structure [3]:

```
Cert-Gen-Schema DEFINITIONS AUTOMATIC TAGS ::=
BEGIN
    EmployeePart ::= SEQUENCE
    {
        signatureEmployee   Signature,
        publicKey           PublicKey
    }

    MediatorPart ::= SEQUENCE
    {
        signatureMediator   Signature,
        publicKey           PublicKey
    }

    Signature ::= SEQUENCE
    {
        signature   INTEGER,
        randomness  INTEGER
    }

    PublicKey ::= SEQUENCE
    {
        primeMOdulus    INTEGER,
        primeDivisor    INTEGER,
        generator       INTEGER,
        publicKey       INTEGER
    }


    CertificateResponse ::= SEQUENCE
    {
        cert    Certificate
    }

     Certificate  ::=  SEQUENCE
     {
        tbsCertificate       TBSCertificate,
        signatureAlgorithm   AlgorithmIdentifier,
        signatureValue       BIT STRING
    }

    TBSCertificate  ::=  SEQUENCE
    {
        version                 INTEGER  {  v3(0)  },
        serialNumber            CertificateSerialNumber,
        signature               AlgorithmIdentifier,
        issuer                  Name,
        validity                Validity,
        subject                 Name,
        subjectPublicKeyInfo    PublicKey,
        subjectUniqueID         UniqueIdentifier,
        extensions              Extension
    }

    CertificateSerialNumber  ::=  INTEGER

    Validity ::= SEQUENCE
    {
        notBefore      Time,
        notAfter       Time
    }

    Time ::= CHOICE {
         utcTime        UTCTime,
         generalTime    GeneralizedTime
    }

    UniqueIdentifier  ::=  BIT STRING

    Extension  ::=  SEQUENCE
    {
        extnID      OBJECT IDENTIFIER,
        critical    BOOLEAN DEFAULT FALSE,
        extnValue   OCTET STRING
                    -- contains the DER encoding of an ASN.1 value
                    -- corresponding to the extension type identified
                    -- by extnID
    }

    AlgorithmIdentifier  ::=  SEQUENCE
    {
        algorithm               OBJECT IDENTIFIER,
        parameters              ANY DEFINED BY algorithm OPTIONAL
    }
END
```

4.2 Signature Generation

The signature generation procedure described in **Chapter 2** uses four messages that are sent between the mediator and the employee. Those messages will be referred to as `EmployeeCommit`, `MediatorCommit`, `EmployeeResponse` and `MediatorResponse` respectively. Below we provide the ASN.1 type specification for all three messages of the signing procedure along with the type of the resulting signature:

```
Sig-Gen-Schema DEFINITIONS AUTOMATIC TAGS ::=
BEGIN
    EmployeeCommit ::= SEQUENCE
    {
        eEphHash    OCTET STRING
    }

    MediatorCommit ::= SEQUENCE
    {
        eEphHash    OCTET STRING
        mEphemeral  INTEGER,
    }

    EmployeeResponse ::= SEQUENCE
    {
        eEphemeral  INTEGER,
        mEphemeral  INTEGER,
        inHMAC      OCTET STRING
    }

    MediatorResponse ::= SEQUENCE
    {
        mSignature  INTEGER
    }

    Signature ::= SEQUENCE
    {
        signature   INTEGER,
        randomness  INTEGER
    }
END
```

References

[1] Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile, 2019.

[2] Yevgeniy Dodis Antonio Nicolosi, Maxwell Krohn and David Mazieres. Proactive two-party signatures for user authentication, 2003.

[3] Sharon Boeyen, Stefan Santesson, Tim Polk, Russ Housley, Stephen Farrell, and David Cooper. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, May 2008.

[4] Joe Hurd. Verification of the miller–rabin probabilistic primality test. *The Journal of Logic and Algebraic Programming*, 56(1-2):3–21, 2003.

[5] NIST. Fips_186 digital signature standard. u.s. department of commerce/n.i.s.t., national technical information service, 1994.

[6] NIST. Fips_186_4 digital signature standard. u.s. department of commerce/n.i.s.t., national technical information service, 2013.

[7] NIST. Transitioning the use of cryptographic algorithms and key lengths, 2019.

[8] J Shawe-Taylor. Generating strong primes. *Electronics Letters*, 22(16):875–877, 1986.