Secure and Privacy-Friendly Public Key Generation and Certification

Fábio Borges*, Leonardo A. Martucci†, Filipe Beato‡, and Max Mühlhäuser*
*Technische Universität Darmstadt, CASED – Telecooperation Lab
DE-64289 Darmstadt, Germany
fabio.borges@cased.de, max@informatik.tu-darmstadt.de
†Karlstad University
SE-651 88 Karlstad, Sweden
leonardo.martucci@kau.se
‡K.U. Leuven – ESAT/COSIC and iMinds
B-3001 Leuven-Heverlee, Belgium
filipe.beato@esat.kuleuven.be

***Abstract*—Digital societies increasingly rely on secure communication between parties. Certificate enrollment protocols are used by certificate authorities to issue public key certificates to clients. Key agreement protocols, such as Diffie-Hellman, are used to compute secret keys, using public keys as input, for establishing secure communication channels. Whenever the keys are generated by clients, the bootstrap process requires either (*a*) an out-of-band verification for certification of keys when those are generated by the clients themselves, or (*b*) a trusted server to generate both the public and secret parameters. This paper presents a novel constrained key agreement protocol, built upon a constrained Diffie-Hellman, which is used to generate a secure public-private key pair, and to set up a certification environment without disclosing the private keys. In this way, the servers can guarantee that the generated key parameters are safe, and the clients do not disclose any secret information to the servers.**

***Index Terms*—Public Key Generation, Privacy, Security, Certification, Protocol.**

I. Introduction

Electronic services permeate the fabric of today's society, which results in an increase need for communication between parties. The setting up of these communication channels has many network-related requirements. In this paper, we address a fundamental security and privacy requirement: the generation and certification of public keys, which are the basis for establishing secure communications channels.

A well-known method for two communicating parties to agree on a symmetric shared encryption key, which is used to establish a secure communication channel, is to perform a secret handshake based on the Diffie-Hellman (DH) key exchange protocol [1]. DH is vulnerable to *man-in-the-middle attacks*, i.e., the communicating parties have no guarantees that their communication channel is not tampered. To perform an authenticated DH key exchange protocol, both parties are required to have their public keys certified and available for verification, i.e., vouched by a trusted third party.

Internet services generally use a certificate enrollment protocol based on standards like PKCS#11 [2], that requires an out-of-band verification channel for identity verification, e.g., email. In this case, the key pair is generated on the client side and then a proof is sent to the server for verification. A second method to provide certification is for a trusted server to generate the key pairs and to deliver the public and secret parameters to its clients (end users), who may have their identity proven using, again, an out-of-band protocol. This is generally the case for electronic identity cards (eID) and smart meters (for the Smart Grid). Both methods present security and privacy issues. While, the first requires an extra channel for verification the later requires a trusted server and may also require an out-of-band verification. Furthermore, the second method requires the key generation server to be fully trusted, as it has fully knowledge of the private key.

Recently, the trusted computing group (TCG) outlined an Endorsement Key (EK) Credential and Platform Credential enrollment standard for TPMv1.2 [3]. The EK is embedded in the Trusted Platform Module (TPM) security hardware, usually at the time of manufacture. In the standard the TCG highlights the importance of the EK to be more constrained than a general RSA private key, i.e., it can only be accessed by the TPM to decrypt specific structures. However, it is important to avoid insecure parameters along with some sort of certification.

Boneh and Matthew [4] showed how parties can share a RSA key without any of them knowing the prime numbers. Distributed processing may be used to ensure that the TPM generated secure parameters. However, it may also leak information about the secret key. Further, Boneh and Franklin [5] and Gilboa [6] present two-party distributed protocols for generating RSA keys. However, their protocols allow both communicating parties to have knowledge of the private keys.

In this paper, we introduce a new paradigm on public key certification, where the key is generated in a collaborative manner with the certification entity. Hence, we present a practical and the first, to the best of our knowledge, constrained key agreement protocol. We follow the definition of key agreement as given in [7], i.e., it is a key exchange in which two or more entities equally contribute toward the computation of the resultant shared key values. Thereby, we define a constrained key agreement to be a key agreement protocol that is achieved in collaboration between the interested parties, and is subject to constraints, such as the prevention of insecure parameters from being chosen in the key exchange. Our novel scheme allows for public key generation in collaboration with the certification authority without the need of an extra out-of-band verification. It also prevents the selection of insecure parameters in the key agreement. For instance, small prime exponent is usually used to improve the performance of RSA [8] as modular exponentiation using large exponentiations is a resource demanding operation [9]. However, the selection of short exponents can lead to security vulnerabilities [7], [10].

a) ***Contributions****:* We present a novel paradigm for key agreement and generation without extra certification verification. The constrained key agreement protocol allows parties to collaborate on the generation of a unique public-private key pair, along with an authority, without disclosing the private part. Our protocol is based on the DH assumption.

b) ***Outline****:* Section II presents the motivation for our work and applications scenarios. In Section III, we review a key agreement protocol that is used as a server-aided key generation protocol. In Section IV, we introduce the preliminaries to our work. Our cryptographic construction is presented in Section V, which is followed by a practical evaluation of our proposal in Section VI. Section VII concludes the paper.

II. Application Scenarios

The motivation for this work comes from security and privacy-preserving applications for the Smart Grid. In addition, our proposed protocol is also applicable in other scenarios, such as electronic identity cards.

A. Electronic Identity Cards

National identity cards in the European Union (EU) are now mostly electronic. EU citizens can sign digital documents using a private key stored in their electronic identity cards (eID). An EU citizen obtains an eID with a public-private key pair already stored in it. This setting is not optimal from the computer security perspective, as there is no strong guarantee that the eID holder is the only one that possesses the eID's private key. The reason behind this is that eIDs are collected at a Registration Authority (RA), which verifies citizens' identities, and the eID is produced by a Certificate Authority (CA) [11]. EID holders are the ultimate responsible for their public-private key pairs. To increase their protection, the key pairs in an eID are PIN protected, i.e., they can only be accessed with the knowledge of a (short) numerical authentication code.

Ideally, eID holders should generate their key pairs and choose their PIN codes simultaneously, as the key pairs are not PIN protected from their generation until a PIN code is selected. The keys also need to be safe, but literature shows that massive production of new and unique key pairs is a challenge to the pseudorandom algorithms involved in the operation [8]. Furthermore, the RA needs also to be trusted.

Our proposed constrained key agreement protocol can be used in the communication between a CA and eID holders. It allows citizens to generate unique and safe key pairs.

B. The Smart Grid

The Smart Grid can be shortly described as a computerized electrical power grid that can provide benefits and services to utilities and end-consumers, such as micro-generation and dynamic pricing for end-consumers, which are not possible in non-computerized power grids. Economic and environmental benefits to the society include, for instance, lowering the ceiling for power generation total capacity as dynamic pricing schemes would tend to flat out peaks in power consumption. The Smart Grid is, however, an intrusive technology. The smart meters that collect power consumption information also allow utilities to profile the habits of their end-customers, i.e., when they are at home or not, when and which appliance is turned on and off, and which TV program is being watched [12].

Preserving end-consumers privacy in the Smart Grid is a research problem that has attracted significant attention of the academic community and a number of privacy-enhancing protocols have already been proposed, e.g., [13]–[15]. A significant share of these protocols require the smart meters to communicate with each other and with the utility. A CA is usually needed to certify public keys. The CA signs a public key and sends the signed key to the smart meter. In addition, every smart meter should generate a safe and unique key pair.

C. Insecure Cryptographic Parameters

The selection of insecure cryptographic parameters can drastically reduce the level of security and usefulness of security protocols. For instance, two entities running a key agreement protocol need to choose a group over an elliptic curve. Restrictions on the curve selection include avoiding anomalous and supersingular curves. For instance, supersingular curves result in a reduction that provides a probabilistic subexponential time algorithm for solving the Discrete Logarithm Problem (DLP) [16].

A more practical example on the selection of insecure cryptographic parameters came from Lenstra *et al.* [8]. They show that more than 95% of the public exponent in the Internet is `65537`. They also show that the probability to find an insecure public key in the Internet is one in five hundred. Our proposed protocol can be used to enforce the randomization of a public key or enforce finding a $n = pq$ such that n starts, contains, or finishes with some sequence of digits. In summary, a constrained key agreement can be used to avoid the selection of insecure parameters.

III. Preliminaries

In this section, we present the preliminaries related to our proposed key agreement protocol as elaborated throughout this paper. Without loss of generality, we consider that an entity A wants to be connected and certified by an entity B. The entity A can be seen as a user that wishes to create an account for a service offered by B, which does not trust A with the generation of its secret factors. Further, we assume that the two parties are online when they perform the key enrollment protocol and do it over a secure channel. In addition, it should be hard for B to compute the private key of A. We define privacy-friendly public key generation and certification as the setup of a public key and its certificate with computations of both A and B where B ensures that A has chosen safe parameters without A disclosing secret information.

Key Agreement: A key agreement protocol is a cryptographic method that allows two entities, A and B to collaborate and agree on a shared secret. The most common cryptographic method for two entities to agree on a secret value is to use the Diffie-Hellman (DH) key agreement [1] over $\mathbb{Z}_p$. The entity A sends $k_A \equiv g^a \mod p$ to B and B sends $k_B \equiv g^b \mod p$ to A. Hence, both A and B can compute $K \equiv (k_B)^a \equiv (k_A)^b \mod p$. Thus, K is the shared secret between A and B. Usually, the shared secret K is used as the secret key for a symmetric encryption algorithm to establish a secure channel. However, K can also be used by both parties to agree on a public key. Note that the value of K is determined by A and B in a collaborative manner. In this way, both parties can secure their communication from an external party by using K.

RSA Key Agreement: To the best of our knowledge, the work in [17] is the unique that presents an improvement over the DH designed to test for invalid RSA public keys. The related algorithm differs from the DH on two factors. First, it works over $\mathbb{Z}_n$ instead of $\mathbb{Z}_p$. Second, it verifies if the public key is valid by evaluating $\gcd(e, \varphi(n)) \neq 1$, where φ is Euler's totient function. Note that this algorithm allows A and B to agree on a RSA public key e, where the private key d is only known by A. The algorithm is depicted by Algorithm 1 and its security is based on the discrete logarithm problem (DLP), i.e., it is hard for B to learn r from $(k^r)^s$.

Algorithm 1: DH for an RSA public key [17]

```
begin
    A chooses two primes $p$ and $q$, and computes $n = pq$
    A chooses $1 < k \in \mathbb{Z}_n$ such that $\gcd(k, n) = 1$
    A sends $(k, n)$ to B
    A chooses $r \in \mathbb{Z}_n$
    A calculates $k^r$ and sends the result to B while
    keeping $r$ secret
    B chooses $s \in \mathbb{Z}_n$
    B calculates $k^s$ and sends the result to A while
    keeping $s$ secret
    Both A and B calculate $e = (k^r)^s = (k^s)^r$
    if $\gcd(e, \varphi(n)) \neq 1$ then
        A sends false to B
        A runs goto begin
    else
        A sends true to B
        A calculates $d = e^{-1}$
```

IV. Our construction

In this section, we present the construction of our constrained key agreement protocol. The result of the construction is presented in the end of this section in Algorithm 2.

Only A can compute d, since only A knows the values of p and q. For the same reason, only A can compute $n = pq$ and $\varphi = \varphi(n) = (p-1)(q-1)$ and only A can verify that e is correct. Note that if one of the primes is known, i.e., if an entity C can solve the DLP or the integer factorization problem, it can calculate φ and, therefore, the inverse exponent $d = e^{-1}$.

The efficient key agreement protocol subject to constraints proposed in this paper builds upon the DH key agreement to establish an RSA key, i.e., it satisfies the restriction

$$\gcd(e, \varphi(n)) = 1. \tag{1}$$

Let U be a multiplicative group formed by the ring $\mathbb{Z}_n$, i.e.,

$$U = \{x \in \mathbb{Z}_n | \gcd(x, n) = 1\}. \tag{2}$$

If $k \in U$, therefore $\gcd(k, n) = 1$ from (2), then $(k^s)^r$ does not belong to U for all values of r and $s \in \mathbb{N}$. Because we can write

$$(k^s)^r = \underbrace{k \cdots k}_{rs = a\varphi(n) + b} \tag{3}$$

and $k^\varphi = 1$, since φ is the order of U, therefore from (3), we have

$$(k^s)^r = k^{a\varphi + b} = k^b, \tag{4}$$

for some a and $b \in \mathbb{N}$. Notice that if $rs < (p-1)(q-1)$, then $a = 0$. Thus, the restriction $\gcd((k^s)^r, \varphi) = 1$ is not always satisfied. Appropriate values for k^s and r need to be chosen. These values are in the range between 2 and $\varphi - 1$. This range contains values that are not co-prime with φ, i.e., that do not satisfy the restriction (1). However, a valid exponent r from (3) can be searched until a valid b for (4) is found, i.e., that satisfies the restriction (1).

For example, consider $p = 3$ and $q = 5$, if we choose $k = 2$, then we have $k | \varphi$. However, $k^4 \equiv 1 \mod pq$ and for all exponents r, we have

$$\gcd(k^r, \varphi) = 2^{(r \mod 4)},$$

i.e., every exponent r that is not of the form $4t$, where $t \in \mathbb{N}$, provides results with some factor in common with φ. However, if $\gcd(k, \varphi) \neq 1$ then $\gcd((k^r)^s \mod pq, \varphi) \neq 1$. Therefore, $\gcd(k, \varphi) = 1$ is a necessary condition of restriction (1), but it is not a sufficient condition of $\gcd((k^r)^s \mod pq, \varphi) = 1$. To illustrate the necessary condition, we consider the following numerical example. If $k = 12$, then $k^2 \equiv 9 \mod 15$ and $\gcd(9, 8) = 1$. To illustrate the $\gcd(k, \varphi) = 1$ is not sufficient condition for $\gcd((k^r)^s \mod pq, \varphi) = 1$, we consider the following numerical example. If $k = 7$, then $k^2 \equiv 4 \mod pq$ and $\gcd(k^2 \mod pq, \varphi) = 4$. Not every exponent rs of the form $2t$, where $t \in \mathbb{N}$, provides a valid exponent. However, if we choose the next exponent k^3, then $k^3 \equiv 13 \mod pq$ and $\gcd(k^3, \varphi) = 1$.

Therefore, an iterative procedure that searches for valid exponents can be designed. Such an iterative procedure can prevent unnecessary selection of new exponents and unnecessary data retransmission between A and B. The constrained key agreement, which includes the iterative procedure, is presented in Algorithm 2. The algorithm checks for invalid RSA exponents before to send k^r. It thus does not require all values to be chosen again and then retransmitted.

Algorithm 2: Constrained Diffie-Hellman on RSA

```
1  begin
2      A chooses two safe primes $p$ and $q$, and computes $n = pq$
3      A chooses $k < n$ with big order.
4      A sends $n$ and $k$ to B
5      B chooses a randomized $s \in \mathbb{Z}_n$
6      B calculates $k^s$ and sends the result to A while keeping $s$ secret
7      $e = 0$
8      while $\gcd(e, \varphi(n)) \neq 1$ do
9          A chooses randomized $r \in \mathbb{Z}_n$
10         A calculates $e = (k^s)^r$
11     A calculates $k^r$ and sends the result to B while keeping $r$ secret
12     B calculates $e = (k^r)^s = (k^s)^r$
13     A calculates $d = e^{-1}$
```

The key agreement protocols that implement the Algorithm 1 and 2 are presented in Figure 1. They show a key agreement procedure of two parties A and B and the data payload exchanged in each step of it.

Figure 1 illustrates that a key agreement protocol implemented using the Algorithm 1 requires one extra message to be exchanged at least. If the restriction of Algorithm 1 is not satisfied, all messages need to be retransmitted (and new values chosen). This procedure is repeated while the restriction is not satisfied. The constrained key agreement protocol implemented using Algorithm 2 always requires only 3 messages to be exchanged between A and B.

Beside the communication, the Algorithm 2 requires an iteration from line 8 to 10, when the Algorithm 1 requires an iteration from line 1 to 12.

The key agreement protocol implemented using Algorithm 1 requires a Boolean value to be transmitted to inform if the restriction was satisfied or not. Algorithm 2 verifies if the exponent values are valid before sending k^r. Therefore, the constrained key agreement reduces the amount of messages exchanged between A and B. It also reduces the number of modular exponentiations required.

V. Analysis

In this section, we analyze our protocol regarding to correctness, complexity, and security.

A. Correctness

The `goto` step of Algorithm 1, which returns the procedure to its beginning if the restriction is not satisfied, results into choosing all values again and the recalculation of all exponentiations. In the proposed Algorithm 2 a `while` procedure is used to choose r and to compute a single exponentiation until the restriction is satisfied. In both Algorithm 1 and 2, only A can verify if the restriction $\gcd(e, \varphi(n)) = 1$ is satisfied or not.

For simplicity, Algorithm 1 and 2 omit the fact that the exponents r and s cannot be a multiple of the order of the element $k \in \mathbb{Z}_n$ as it leads to $e = 1$ (since $k = 1^r$ and then $e = (k^r)^s = 1$).

It is important to notice how big the order of k is. The order o of a primitive roots of unity k in $\mathbb{Z}_n$ is divisors of $\lambda(n)$, i.e., $o|\lambda(n)$, where $\lambda(n)$ is a generalization of $\varphi(n)$ and is defined as the smallest positive integer o such that $k^o = 1$ for ever integer k that is coprime to n. For an analysis of the average and minimal order of $\lambda(n)$, see [18]. The function $\lambda(n)$ can be defined as $\lambda(1) = \varphi(1) = 1$; and

$$\lambda(p^v) = \begin{cases} p^{v-1}(p-1) = \varphi(p^v) & \text{if } p \geqslant 3 \text{ or } v \leqslant 2, \\ 2^{v-2} = \frac{\varphi(p^v)}{2} & \text{if } p = 2 \text{ and } v \geqslant 3, \end{cases}$$

for a prime p and positive integers v and $n \geqslant 2$; and

$$\lambda(n) = \operatorname{lcm}(\lambda(p_1^{v_1}), \ldots, \lambda(p_i^{v_i})),$$

when $n = p_1^{v_1} \ldots p_i^{v_i}$ is the prime factorization of n. Euler's theorem states that $k^{\varphi(n)} = 1$ for k and n coprime, thus $\lambda(n)$ divides $\varphi(n)$ and o divides $\varphi(n)$. For this reason, we choose safe primes in Algorithm 2, they are prime numbers of the form $2p' + 1$ where p' is also prime. Therefore, from two safe prime numbers p and q, we have

$$\lambda(pq) = \operatorname{lcm}(\lambda(p), \lambda(q)) = \operatorname{lcm}(2p', 2q') = 2p'q', \quad (5)$$

and the only possible values for o are $\{1, 2, p', q', 2p', 2q', p'q', 2p'q'\}$, thus either k has order less than 3 or has big order.

B. Complexity Analysis

The set U from (2) is also denoted $\mathbb{Z}_n^\times$ since $\mathbb{Z}_n^\times \subset \mathbb{Z}_n$. Every RSA key is indeed in the subset $\mathbb{Z}_n^\times$. Thus, the number of valid keys is given by $\varphi(n)$. Therefore, the probability of randomly choosing an invalid key I in n possible values is given by

$$Pr[I] = 1 - \frac{\varphi(n)}{n}. \quad (6)$$

Equation (6) is the probability of Algorithm 1 and 2 selecting an invalid key I. In such an event, Algorithm 1 returns to its initial step, i.e., it goes to `begin` and Algorithm 2 remains in the `while` loop. Assuming that those values are randomly chosen then the events are independent by definition. Hence, the independent probability from (6) remains the same for every interaction.

Let t be an arbitrary number of iterations. Thus, the probability of having t iterations that results in invalid keys I is given by

$$\begin{aligned} Pr[I|t] = Pr[I] & + (1 - Pr[I]) \cdot Pr[I] \\ & + (1 - Pr[I])^2 \cdot Pr[I] \\ & \vdots \\ & + (1 - Pr[I])^t \cdot Pr[I]. \end{aligned} \quad (7)$$

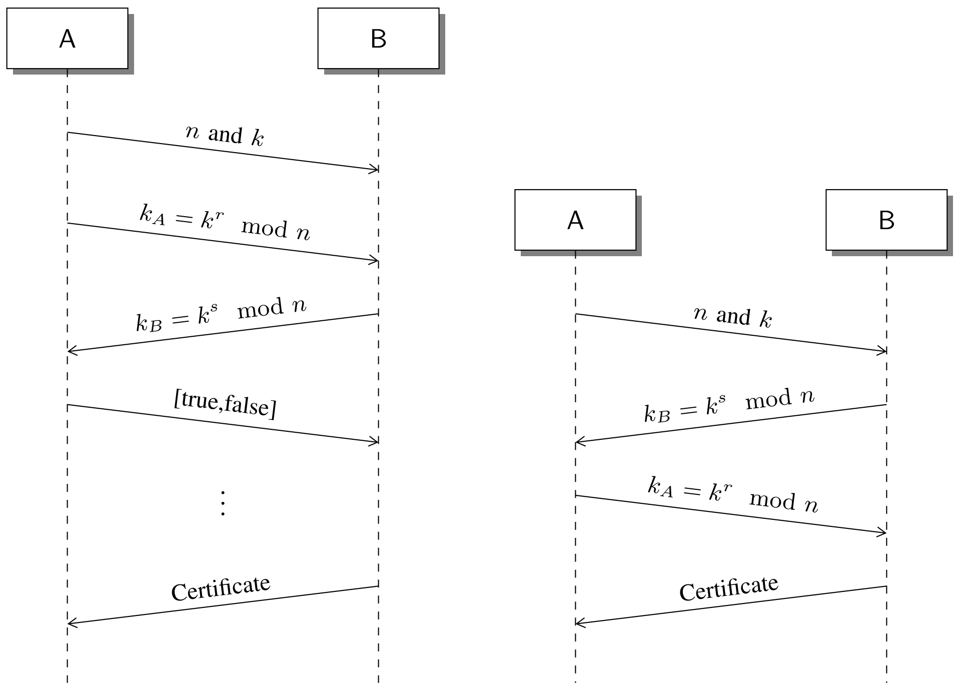

Figure 1: The key agreement protocols between two entities A and B implemented using the Algorithm 1 (left) and Algorithm 2 (right).

The equation (7) can be rewritten as

$$\begin{aligned} Pr[I|t] &= \sum_{i=0}^{t} (1 - Pr[I])^i \cdot Pr[I] \qquad (8) \\ &= \frac{Pr[I] \cdot \left(1 - (1 - Pr[I])^{t+1}\right)}{Pr[I]}. \end{aligned}$$

From (8) it is possible to calculate the complexity of Algorithm 1 and 2. The complexity is calculated in terms of number of modular exponentiations, which represent the most relevant and expensive operations. In both algorithms, entities A and B compute at least two exponentiations each. Therefore, in the best case, both algorithms have the same complexity, i.e., they use the same amount of processing time. However, on the average case, the complexity of both algorithms is different.

To demonstrate this, let E be the cost of a modular exponentiation. Thus, the Algorithm 1 has complexity $\mathcal{O}(4E + 4E \cdot Pr[I|t])$ while Algorithm 2 has complexity $\mathcal{O}(4E + E \cdot Pr[I|t])$. In Algorithm 2, B always computes two exponentiations, i.e., $\mathcal{O}(2E)$, and A computes two exponentiations plus one based on the probability $Pr[I|t]$ (8), i.e., $\mathcal{O}(2E + E \cdot Pr[I|t])$. In Algorithm 1, both A and B have the same complexity $\mathcal{O}(2E + 2E \cdot Pr[I|t])$, which depends on the probability $Pr[I|t]$. Therefore, Algorithm 2 is $\mathcal{O}(3E \cdot Pr[I|t])$ efficient than Algorithm 1.

C. Security

As aforementioned, the goal is to protect the private part of the key from the server, i.e., B. Thereby, it should be hard for B to compute or learn d. For this, the Algorithm 2 relies on the DLP, as it is hard for B to compute r by having k^s. Notice that B can verity if n is a product of safe primes without A discloses their values [19]. Hence, B can also verify if k has high order.

Besides the generation of keys based on constrain, the security of the key agreement is the same as the DH. We now turn and evaluate the generation of keys based on constrain.

1) Key Space: The requirement of the privacy-friendly public key generation and certification protocol is that all operations lie over the ring $\mathbb{Z}_n$ and, thus, all keys need to be over the multiplicative group $\mathbb{Z}_n^\times$ to be valid.

First, we need verify if the probability of Algorithm 2 finds a valid key K is acceptable. We know that the probability to find a valid key is

$$Pr[K] = \frac{\varphi(n)}{n} = \frac{pq - p - q}{n} = 1 - \frac{p+q}{n}.$$

Let $\log_2(n) \approx N$, therefore $\log_2(p+q) \approx \frac{N}{2}$ and

$$Pr[K] \approx -\frac{1}{2^{2/N}}.$$

Thus, the probability to find a valid key tends to 1 when N tends to ∞. Moreover, for the minimum acceptable N in the RSA we have

$$Pr[K] \approx 1 - \frac{1}{2^{512}}.$$

Thus, the Algorithm 2 can find a valid key with very high probability.

2) Probability to Find a Unique Key: Peeking randomly valid key K, the independent probability to peek K again is

$$P[K,K] = \left(\frac{1}{\varphi(n)}\right)^2,$$

therefore,

$$P[K,K] \approx \frac{1}{N^{2N} - 2\,N^N N^{1/2\,N} + N^N}$$

For all N greater than 2, we have

$$\frac{1}{N^{2N}} < P[K,K] < \frac{1}{N^N}.$$

As the probability to find a different key is $1 - P[K,K]$, then the Algorithm 2 can find a unique key with very high probability .

3) Probability to Find a Safe Key: Let E be the number of bits to represent an exponent, then the probability to find a key K with length E is

$$Pr[K_E] = \frac{2^E}{\lambda(n)}$$

and $\log_2(\lambda(n)) \approx N - 1$, therefore,

$$Pr[K_E] \approx \frac{1}{2^{N-1-E}}. \tag{9}$$

When E tends to N, the probability tends to 1, and when E tends to zero, the probability tends to a very small ϵ close to zero. Since we chose safe primes, from equation (5), we have maximum values to exponents e and d are either very small or very big. Nevertheless, equation (9) shows that very small exponent is highly improbable.

Here an important point may arise: what a safe key is. For save processing time many software use the public key $65\,537 = 2^{16} + 1$ [8], but the private key is generated by the Euclidean algorithm, which can be assumed as randomly. In this case, the key size has the same probability than Algorithm 2. Nevertheless, the Algorithm 2 generates randomly both key.

As aforementioned small keys should be avoided [7], [10], but today it is very common the use of a key with 16 bits. Algorithm 2 generates much large keys with very high probability.

D. Man-in-the-middle attack

Algorithm 2 was developed to enforce the generation of unique and safe keys, and should run on a secure channel of communication. However, if a third entity perform a man-in-the-middle attack pretending be A to B and B to A then at the end of the algorithm A will not receive the certification of B. In the case of the smart metering scenario, the smart meter will not receive the valid certificate.

VI. Practical Evaluation

To demonstrate the practical feasibility and efficiency of our protocol, we have performed a practical evaluation based on simulations. In addition, we compared with the Algorithm 1. Thereby, we developed two client-server applications that implement the two key agreement protocols presented in this work, i.e., one implements the Algorithm 1 and the other the Algorithm 2. The applications were implemented in `Python` version 2.7.1+ [GCC 4.5.2] using `gmpy` – Multiprecision arithmetic for Python version 1.14 as library. The applications were running on an Intel Core(TM)2 Duo CPU T9400 2.53GHz processor with 4GB of memory. The operating system used was Windows 7 Enterprise 64 bits as host and two identical virtual machines Linux Mint version 2.6.38-8-generic for 64 bits with 1GB of memory on VirtualBox version 4.1.6. One virtual machine ran the server program and the other the client program. Both virtual machines communicated directly over their virtual network interfaces and were isolated from any external network traffic. The application parameters are chosen using a pseudo-random number generator. The value n has 1024 bits such that the primes p and q have 512 bits each, and other parameters have 1024 bits. The metric used is the processing time required to establish the key. To obtain statistical confidence in our results, 10 000 simulation runs were used for each key agreement protocol.

The outcome of our simulation is depicted in Figure 2, and shows that our constrained key agreement protocol, i.e., Algorithm 2, outperforms the protocol implemented using Algorithm 1. In fact, while the server side on our protocol takes 73ms on average, the key agreement protocol using Algorithm 1 needs 136ms on average to agree on one key. On the client side, our constrained key agreement protocol needs 64ms on average while the protocol using Algorithm 1 requires 130ms on average to agree on one key. Figure 2 also illustrates the variance bars. The standard error is negligible and, thus, omitted from the figure. The figure shows that the server requires on average more processing time than the client in both key agreement protocols. This is an expected fact as servers need to perform more operations per key agreement than the clients, as depicted in Figure 2.

Figure 3 presents the box plot of the total processing time required by both key agreement protocols to agree on one key on the client A and server B sides. The lower, median and upper quartiles are shown in the figure. The average and the smallest and largest observations are also depicted. Figure 3 shows that the upper and lower quartiles are closer to the median in the case of the constrained key agreement, which indicates predictable results regarding the required processing time in comparison to the key agreement protocol using the Algorithm 1. Furthermore, the largest observations for the Algorithm 1 are 1.96s for the server side and 1.97s for the client side, which are considerably higher than the largest observations for the constrained key agreement using the Algorithm 2, which are 1.50s for the server side and 1.52s for the client side.

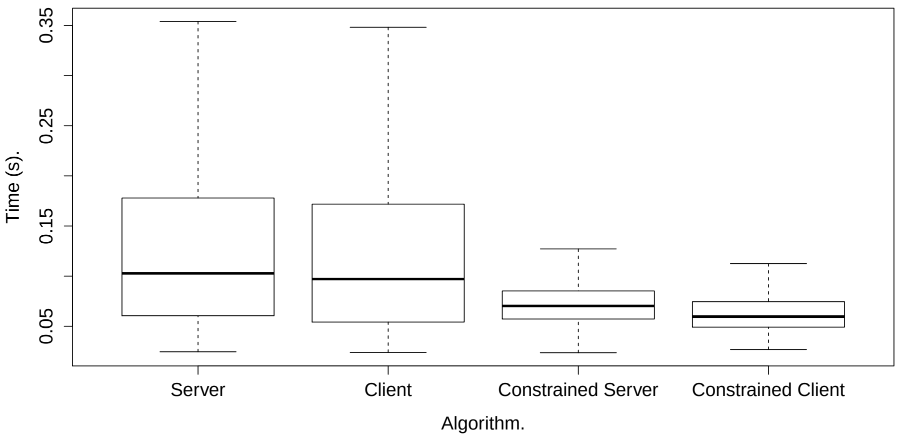

Figure 3: Total processing time required by clients and servers to agree on a key using the key agreement protocol using the Algorithm 1 (left) and using the constrained key agreement that implements the Algorithm 2 (right). The outliers are not shown in the box plot.

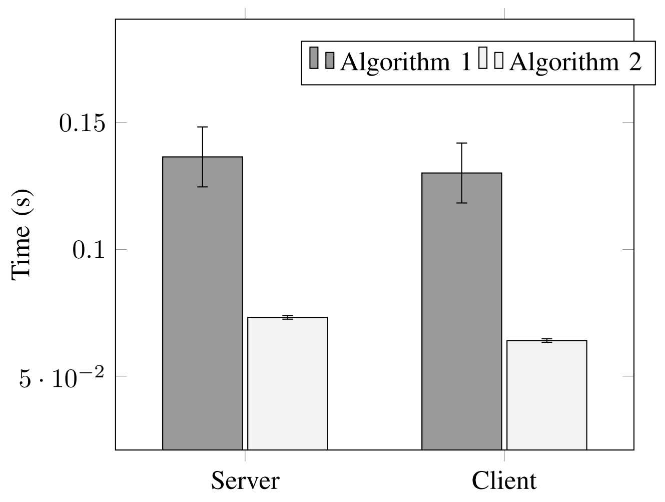

Figure 2: The mean processing time for clients and servers needed to run the key agreement protocol using the Algorithm 1 and the constrained key agreement using Algorithm 2 are compared in this figure.

Moreover, Figure 4 illustrates the box plot of the total number of failed attempts in the client and server sides to agree on one key. Fails are understood differently on the server and client sides. On the server side, a fail means that the server had to choose a new value. In the case of the key agreement protocol using the Algorithm 1, a fail means that the restriction was not satisfied and the algorithm goes to the `begin`. In the case of the constrained key agreement protocol (using Algorithm 2) a fail means that the restriction was not satisfied and the `while` loop continues. On the client side, a fails means that the client has tried to start a run of the key agreement protocol and the server was busy and unable to reply. Servers are considered busy when they have yet not finished the last key agreement session. The lower, median and upper quartiles are shown in Figure 4. The average and the smallest and largest observations are also depicted. Figure 4 shows that the number of fail attempts is practically the same in both key agreements. That was expected, as all values are chosen using a pseudo-random number generator. Hence, the odds for choosing a value that does not satisfy the restriction (in the server B) is the same for both key agreement protocols being evaluated. On the client side, a similar number of failed attempts was expected as the amount of time required for a server to end the protocol is the same on both key agreement protocols, i.e., the time required to close the TCP connection.

In our evaluation, we have measured the required processing time for both key agreement protocols. Our measurements did not consider the end-to-end network delay since that would add an uncontrolled parameter in our evaluation. Hence, to eliminate the influence of such parameters, both client and server were implemented using two virtual machines running on the same host. However, by introducing the end-to-end network delay into our evaluation would further improve our results, because the constrained key agreement protocol requires fewer messages to be exchanged than the DH key agreement for an RSA public key.

VII. Conclusion

In this paper, we have presented a constrained key agreement protocol that is used to setup an initial environment for certification protocol with privacy. The protocol ensures that the key pairs are secure and unique. The constrained key agreement protocol is built upon a constrained DH on RSA algorithm. It uses an iterative procedure that searches for valid exponents and prevents the re-selection of unnecessary values and data retransmissions. Our protocol was evaluated and compared against a DH key agreement for an RSA public

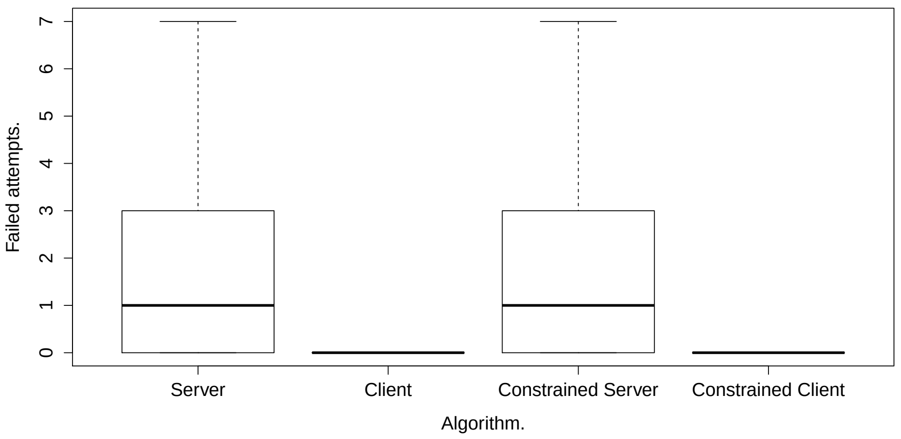

Figure 4: Number of failed attempts of clients and servers to agree on one key using the Algorithm 1 (left) and using the constrained key agreement that implements the Algorithm 2 (right). The outliers are not shown in the box plot.

key (Algorithm 1) [17]. We implemented both key agreement protocols in two client-server applications in order to compare and evaluate both protocols. Our simulation results have shown that our proposal outperforms the DH key agreement for an RSA public key by approximately a factor of 2 regarding the average processing time.

Constrained key agreement protocols prevent insecure parameters from being chosen by pre-defining the restrictions in the key agreement. Thus, multiple application scenarios presenting such restriction can profit from such protocols. Constrained key agreement protocols can also be used in one-to-many applications [20]. For instance, in ad-hoc networks running anonymous authentication protocols would require the key set to belong to $\mathbb{Z}_n^\times$. Hence, the constrained key agreement is needed. In addition, smart meters running a privacy-friendly public key generation and certification protocol would have a similar requirement and, thus, the same need for a constrained key agreement protocol.

Acknowledgments

The authors are thankful to the Horst-Görtz Foundation and SmartSociety, a project of the Seventh Framework Programme for Research of the European Community under grant agreement no. 600854, for funding this work. Filipe Beato is funded by the FCT PhD fellowship SFRH/BD/70311/2010.

References

[1] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Information Theory*, vol. IT-22, no. 6, pp. 644–654, 1976.

[2] RSA Laboratories, "PKCS #11: Cryptographic token interface standard v2.30," 2009.

[3] Trusted Computing Group, "Infrastructure cmc profile for ek/platform certificate enrollment for tpmv1.2 specification version 1.0," 2013.

[4] D. Boneh and M. Franklin, "Efficient generation of shared rsa keys," *J. ACM*, vol. 48, no. 4, pp. 702–722, Jul. 2001. [Online]. Available: http://doi.acm.org/10.1145/502090.502094

[5] D. Boneh and M. K. Franklin, "Efficient generation of shared rsa keys," *J. ACM*, vol. 48, no. 4, pp. 702–722, 2001.

[6] N. Gilboa, "Two party RSA key generation," in *CRYPTO*, ser. LNCS, M. J. Wiener, Ed., vol. 1666. Springer, 1999, pp. 116–129.

[7] H. C. A. van Tilborg and S. Jajodia, Eds., *Encyclopedia of Cryptography and Security, 2nd Ed.* Springer, 2011.

[8] A. K. Lenstra, J. P. Hughes, M. Augier, J. W. Bos, T. Kleinjung, and C. Wachter, "Ron was wrong, whit is right," Cryptology ePrint Archive, Report 2012/064, 2012, http://eprint.iacr.org/.

[9] P. Lara, F. Borges, R. Portugal, and N. Nedjah, "Parallel modular exponentiation using load balancing without precomputation," *Journal of Computer and System Sciences*, vol. 78, no. 2, pp. 575–582, 2012.

[10] D. Bleichenbacher and A. May, "New attacks on rsa with small secret crt-exponents," in *Public Key Cryptography - PKC 2006*, ser. LNCS. Springer Berlin / Heidelberg, 2006, vol. 3958, pp. 1–13.

[11] G. Aichholzer and S. Strauss, "The citizen's role in national electronic identity management - a case-study on austria," in *Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services, 2009. CENTRIC '09. 2nd Int Conf on*, Sep 2009, pp. 45–50.

[12] U. Greveler, B. Justus, and D. Löhr, "Multimedia content identification through smart meter power usage profiles," in *Computers, Privacy and Data Protection (CPDP 2012)*, 2012.

[13] F. Borges, L. A. Martucci, and M. Mühlhäuser, "Analysis of Privacy-Enhancing protocols based on anonymity networks," in *Smart Grid Communications (SmartGridComm), 2012 IEEE Conference on*, Tainan City, Taiwan, Nov. 2012.

[14] F. Borges, D. Demirel, L. Böck, J. Buchmann, and M. Mühlhäuser, "A Privacy-Enhancing protocol that provides In-Network data aggregation and verifiable smart meter billing," in *19th IEEE Symposium on Computers and Communications (IEEE ISCC 2014)*, Madeira, Portugal, Jun. 2014.

[15] F. Borges and L. A. Martucci, "iKUP keeps users' privacy in the smart grid," in *2014 IEEE Conference on Communications and Network Security (CNS) (IEEE CNS 2014)*, San Francisco, USA, Oct. 2014.

[16] A. Menezes, T. Okamoto, and S. Vanstone, "Reducing elliptic curve logarithms to logarithms in a finite field," *Information Theory, IEEE Transactions on*, vol. 39, no. 5, pp. 1639 –1646, Sep 1993.

[17] R. Klima, N. Sigmon, and E. Stitzinger, *Applications of Abstract Algebra With Maple And Matlab*, ser. Discrete Mathematics And Its Applications. Chapman & Hall/CRC, 2007, no. Bd. 1.

[18] P. Erdös, C. Pomerance, and E. Schmutz, "Carmichael's lambda function." *Acta Arith*, vol. 58, no. 4, pp. 363–385, 1991.

[19] J. Camenisch and M. Michels, "Proving in zero-knowledge that a number is the product of two safe primes," in *Proc of the 17th Int Conf on Theory and application of cryptographic techniques*, ser. EUROCRYPT'99. Springer-Verlag, 1999, pp. 107–122.

[20] J. Chen, R. Boreli, and V. Sivaraman, "Improving the efficiency of anonymous routing for manets," *Computer Communications*, 2011.