LocalPKI: An Interoperable and IoT Friendly PKI

Jean-Guillaume Dumas, Pascal Lafourcade, Francis Melemedjian,
Jean-Baptiste Orfila, Pascal Thoniel

LocalPKI: An Interoperable and IoT Friendly PKI

Jean-Guillaume Dumas[1], Pascal Lafourcade[2], Francis Melemedjian[3], Jean-Baptiste Orfila[1], and Pascal Thoniel[3]

[1] Université Grenoble Alpes, CNRS, Laboratoire Jean Kuntzmann, 700 av. centrale, IMAG, CS 40700, 38058 Grenoble cedex 9, France
{Jean-Guillaume.Dumas,Jean-Baptiste.Orfila}@univ-grenoble-alpes.fr
[2] University Clermont Auvergne, LIMOS, Campus Universitaire des Cézeaux, BP 86, 63172 Aubière Cedex, France
Pascal.Lafourcade@uca.fr
[3] NTX Research SA, 111 Avenue Victor Hugo, 75116 Paris, France
{Francis.Melemedjian,Pascal.Thoniel}@ntx-research.com

Abstract. A public-key infrastructure (PKI) binds public keys to identities of entities. Usually, this binding is established through a process of registration and issuance of certificates by a certificate authority (CA) where the validation of the registration is performed by a registration authority. In this paper, we propose an alternative scheme, called LOCALPKI, where the binding is performed by a local authority and the issuance is left to the end user or to the local authority. The role of a third entity is then to register this binding and to provide up-to-date status information on this registration. The idea is that many more local actors could then take the role of a local authority, thus allowing for an easier spread of public-key certificates in the population. Moreover, LOCALPKI represents also an appropriate solution to be deployed in the Internet of Things context. Our scheme's security is formally proven with the help of Tamarin, an automatic verification tool for cryptographic protocols.

1 Introduction

The primary goal of a *Public Key Infrastructure* (abbreviated *PKI*) is to bind a user identity with his public key. They are security architectures that manage *digital certificates*, electronic documents used to prove the ownership of a public key. The current global PKI standard is *PKIX*.

Deploying server certificates is a necessity to bring trust in transactions made on the Internet. This also mandatory to be able to use electronic signatures, for authentication, session key transport, authenticated key exchange, or more generally, any secured communication.

Started near the end of 2015, the Let's Encrypt project goes further and seeks to democratize the implementation of server certificates. The goal is to generalize certificates for almost all servers. Why such an initiative? Two constraints have been identified: the cost of certificates and the complexity of their implementation. Let's Encrypt provides a solution to remove these bottlenecks and as of mid-2018 this has become the choice of more than half of the market shares.

Now, the deployment of certificates for persons (citizen, professional, student, consumer, Internet user) is also a necessity to bring trust in transactions made on the Internet. Despite *PKIX*, this deployment has never really been carried out on a large scale and few of us now possess these certificates. The first goal of LOCALPKI is to democratize the attribution of people certificates, just like Let's Encrypt democratized the attribution of server certificates. The second goal of LOCALPKI is enable an easier deployment of public key certificates in constrained environments like the IOT *Internet of Things*.

Overall, the goal is to generalize certificates for all. How? By removing the three known bottlenecks with *PKIX*: remote delivery, cost of certificates and their complexity of use. LOCALPKI provides a solution on organizational, financial and cryptographic levels (new protocols) in order to eliminate the current, above listed, bottlenecks/issues. For this, the central actors in LOCALPKI will be the Local Registration Authorities (LRA). Those will be established industrial actors, like a bank or a telecom operator for instance, and they will enroll users who are their members or customers thanks to their local network. The Local Registration Authority will remain close to its users, it will have to be the local bank branch office or the next door Telecom agency. Indeed, it is a foremost importance for these actors to deliver to their members or clients certificates that will enable them to authenticate and sign contracts online. Then, to guarantee the security of the system, Electronic Notaries will be in charge of maintaining the data bases of registered certificates.

Now, for the user, he/she can get a free certificate (with paid options available) near his/her home or work place with the security of face-to-face enrollment (identity verification). The system can also support trust circles. For example, if a user trusts a given financial or telecom player to enroll its customers, then the user will be convinced by the public key authenticity assertion sent by its Digital / Electronic Notary server about one of its customers. Finally, LOCALPKI is not only limited to personal certificates. Tomorrow, the deployment of connected objects certificates will also be a necessity to bring trust in transactions made on the Internet of Things (IOT, Internet of Things). Each manufacturer will become the Registration Authority for its connected objects.

1.1 State of the art

Over the internet, the main standard for the format of public key certificates is X.509, developed by the IETF PKIX working group. This is the most widely used standard in Internet protocols [9]. RFC 5280, which define how to use X.509 in Internet protocols. For instance, almost all major internet players authentify their servers with certificates via the *TLS* (Transport Layer Security) protocol, basis of the *HTTPS* (Hypertext Transfer Protocol Secure). In this paradigm, end entities (users) rely on *Certificate Authorities* (*CA*) which deliver *X*.509 certificates containing, at least, the user's identity and public key and a validity period. These certificates are then signed by the *CA*. Then, before using the public key of another entity, a user verifies the associated certificate: checking validity period, correctness of signature etc. We talk about the *owner* of a certificate and the *user* of a certificate.

Now, in order to be able to use long enough validity periods, a revocation mechanism has then to be set up. Indeed, the burden of renewing one's certificate need not be repeated too often, but a revocation can however be launched in case of unexpected

events. The most deployed solutions for this are *Certificate Revocation List* (CRL) [9] and *Online Certificate Status Protocol* (*OCSP*) [26]. In the first case, the user sends a validity query of the owner's certificate to an *OCSP* responder (either maintained by the certificate's issuer, or well known to the user). In the second case, lists of revoked certificates must be regularly updated and published into *CRL Distribution Points* (*CDP*). Users access these *CDP* and check that the owner's certificate is not present inside the list. In the end, depending on the obtained certificate status, the owner is authenticated or not by the user.

On the one hand, in terms of various security and efficiency requirements, mostly depending on the environment where the PKI must be set-up (internet, industrial architecture, power limited devices...), the current state of the art for PKIs is quite substantial. In terms of efficiency, solutions to reduce communication and computation costs of the revocation checks arose, such as *H-OCSP* [22] or *Delta-CRL* [9]. Moreover, the needed trustfulness in the *CA* has been intensively studied in order to reduce the impact of malicious behavior. Indeed, in *PKIX*, a compromised *CA* is able to ruin the entire authentication mechanism by delivering illegitimate certificates. Then, solutions based on public logs of *CA*'s action emerged, with for instance *Certificate Transparency* [18]. In this paradigm, certifications are included in append-only log structures: the log maintainer is able to prove that a specific certificate is present into the structure, and that each new added certificate has only extended the previous structure. The *Accountable Key Infrastructure* [16] exploits public-logs for the certificate management and distributes the trust between several entities. In [25], the authors provide a solution combining public revocation and a more largely distributed trust (using users' browsers) with the *Certificate Issuance and Revocation Transparency*.

On the other hand, more recently, the *Attack Resilient Public-Key Infrastructure* (ARPKI) [4] proposes a PKI where clients choose several authorities (*CAs* and log maintainers) involved into the global process. There, each of these authorities supervises the others' behavior: as a consequence, a single non compromised entity is sufficient to prevent attacks. The solution described in [29] with the *Distributed Transparent Key Infrastructure* (DTKI), provides security even in the case where all these entities are corrupted. Notably, the security of ARPKI and DTKI has been formally proven using the automatic cryptographic protocols verification tool Tamarin [19,27]. Most of these advanced solutions provide enhanced security to the cost of some complexity in the procedures. In a practical set up these solutions involve several authorities, which must be involved in the management of technical services (this is particularly true for the management of worldwide append-only logs). Furthermore, these schemes rely on a *CA*'s signature of certificates in order to guarantee the binding between the public key and the identity.

1.2 Contributions

In this paper, we formally describe and prove the security of a public-key infrastructure called LocalPKI, based on the *PKI* 2.0 paradigm [6]. The *PKI* 2.0 project and its instances such as LocalPKI are seeking to democratize the attribution of people certificates, like what *Let's Encrypt* is doing for server certificates. They are essential for

secured transactions: authentication, digital signature, confidentiality. LocalPKI removes three known bottlenecks of *PKIX*: remote delivery, cost of certificates and their complexity of use. The idea is to replace *CA* signed certificates by user self-signed certificates. However, contrary to *PGP* [30], trust is not given by users but by an authority. This authority, a combination of notaries and local actors in our setting, guaranties the binding. Indeed, after registration of a certificate owner by a notary, other users will be able to verify the authenticity of this certificate via a request. For a private verification, a possibility is to send a request to the notary, similar to an *OCSP* procedure: the notary's response depends on the looking up of uniquely recorded legitimate users in his database. For a public verification, a possibility is to provide hashed and signed subsets of the database, similar to NSEC3 [17] (or now NSEC5 [28]) records within DNSSEC [5]. Also, as in *PKIX*, registration and authentication do not need be performed by the same entities. In *PKIX* the former step can be performed by a *Registration Authority*. In LocalPKI, this entity is a *local* RA, known by the notary and close to the user. It can be a technical service, just like a classical registration authority, but it can also be a local actor or a business service, closer to users. We have in mind banks, postal offices, mobile network operators, delivery points, university offices, for example, and more generally any actor used to check identities. For the user, he can get a free certificate (with possibly paid options available) near his home or work place with the security of face-to-face enrollment (identity verification). For local registration desks, it would often be in its own interest to deliver people certificates to its members or clients. These member/client certificates will enable them to authenticate and perform online operations, for instance securely signing contracts. Moreover, the involved entities, except for the notaries, do not really need any technical knowledge. Overall, our proposition, LocalPKI, is a first instance of the PKI 2.0 paradigm offering an alternative to *PKIX*. LocalPKI is able to provide the same services as *PKIX*, from authentication to revocation, via, e.g., cross-certification. But the approach uses instead a user-centric model and does not need a signature by an authority for each certificate. From the simplicity of its setup, we show that LocalPKI also becomes an interesting solution to deploy a PKI in the *Internet of Things* context. Further, in this paper, we also provide a security analysis of all our protocols, using the formal verification tool Tamarin [19,27]. Finally, we show that the implementation of the LocalPKI in a practical environment can be effectively realized using existing *PKI* tools: we have deployed a prototype web-based implementation available for testing there: `http://hpac.imag.fr/localpki`.

1.3 Organization of the paper

In section 2, we start by defining entities involved in LocalPKI, and we make a high-level comparison with *PKIX*. Then in Section 3, we formally describe all the protocols involved, i.e. the registration, authentication and revocation mechanisms. We also describe a private blockchain based solution to efficiently interoperate multiple instances of LocalPKI. Section 5 is devoted to the deployment of the architecture, using existing tools and solutions. Finally, in Section 6, we define the required security properties and an associated formal model for LocalPKI. Those enable us to formally prove the security of LocalPKI, using Tamarin. Finally, in Section 7, we show how to deploy LocalPKI as an Internet of Things PKI.

With respect to the conference version of this paper [11], we describe a complete use case of LocalPKI in the Internet of Things context (Section 7). We also extend the system in order to be able to interoperate multiple instances of LocalPKI, for instance using a private blockchain (Section 4). Finally we add some specifications on the deployment and propose a web-based prototype implementation of LocalPKI registration (Section 5).

2 General Description

In this setting, LocalPKI is a set of protocols which fulfills all the requirements of a public-key infrastructure (*PKI*): registration of a new user, authentication of registered users, revocation of certificates, renewals, cross-certification, etc.

Informally, the main idea is that users will produce themselves their self-signed certificates, and that notaries will only store the signed hash of this certificate and its serial number. The notaries them manage these within a database. On-line authentication is realized by the notary, just verifying that a given certificate hash is present or not in the database.

In the following, we give more details on the protocols and start by introducing the different entities and their role.

2.1 Entities

There are three different entities in LocalPKI: the *Electronic Notaries* (*EN*), the *Local Registration Authorities* (*LRA*) and the *users* (or *End Entities*).

The *Electronic Notary* might be seen as the root *CA* in a classical PKI architecture. Actually, he manages the databases containing registered users.

The *Local Registration Authority* represents the intermediate entity between the user and the notary. It is somewhat like a *Registration Authority* in a classical PKI, but in LocalPKI it is closer to users than to the *CA*. In practice, the *LRA* could be an agency close to the user, such as the user's insurance company, his bank, the postal office, etc. Those agencies usually already have the abilities to check identities. The *LRA* is registered by some *EN*, and the identity checks are performed during the recording of a new user. The multiplicity of Registration Authorities reduces the systemic risk that exists in the event of a Certification Authority compromise.

Finally, *users* represent the entities who want to authenticate or be authenticated by others.

2.2 Comparison with *PKIX*

The main differences between LocalPKI and *PKIX* are:

- In LocalPKI registration authorities do not need to be security experts. Therefore they can be closer to users and allow more widespread deployment of the use of certificates in every day life (see § 3.2).
- Certificate creation is done by the *CA* in *PKIX*; in LocalPKI it is done by the *LRA* while the notaries store and make this decision available (see § 3.2).

– The default authentication mode in *PKIX* is a buffering via the *CRL*; while default authentication mode in LOCALPKI is interactive, somewhat like *OCSP Online-Certificate Status Protocol* (see § 3.3).
– The alternative authentication mode in *PKIX* is *OCSP*, while LOCALPKI can also propose an alternative buffering mechanism called *Certificate Verification Lists* (*CVL*), described in Section 3.3.

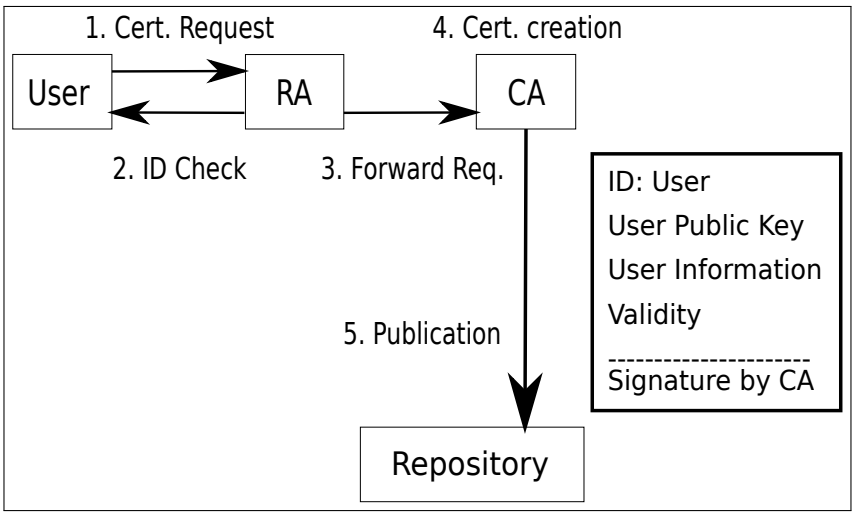

Fig. 1. *PKIX* registration [11, Fig. 2].

From a closer look at both protocols execution, the major difference is in the signature of the user's certificate: in *PKIX*, the *CA*'s signature is present whereas in LOCALPKI only the self signature made by the user is required. As shown in Figure 2, certificate creation is realized in the 2^{nd} step by the owner instead of the 4^{th} one by the *CA* for *PKIX*, as shown in Figure 1. Furthermore, the registration authority forwards a certificate request to the *CA*, while the *LRA* only sends the certificate's hash to the notary. Finally, in LOCALPKI certificates are not directly published since the *EN*'s database only contains hashes.

Hence, interactive authentication is also different as shown in Figures 3 and 4: with LOCALPKI, owners have to provide their certificates to users beforehand, and then the users interact with notaries in order to be convinced of the certificate's validity; with *PKIX* users can recover certificates in a local repository and then interacts with OCSP responders to be convinced of the certificate's validity. The full protocol, in particular the buffering mechanism (the *CVL*, also called public mode within LOCALPKI), is detailed next.

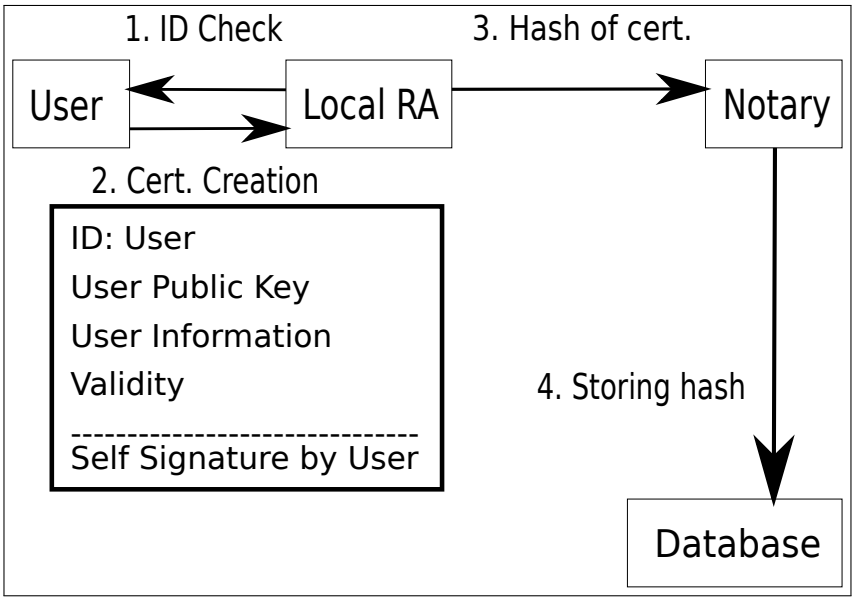

Fig. 2. LOCALPKI registration [11, Fig. 1].

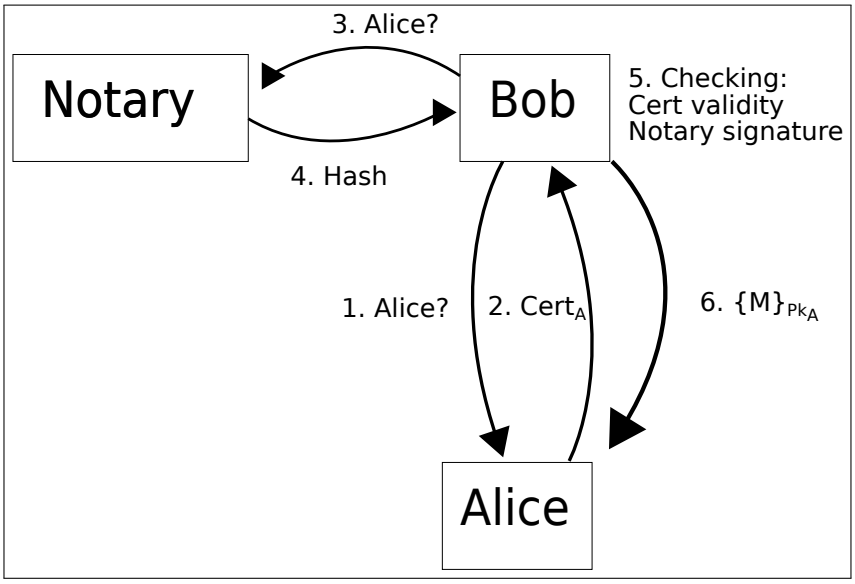

Fig. 3. LOCALPKI end entity interactive authentication [11, Fig. 3].

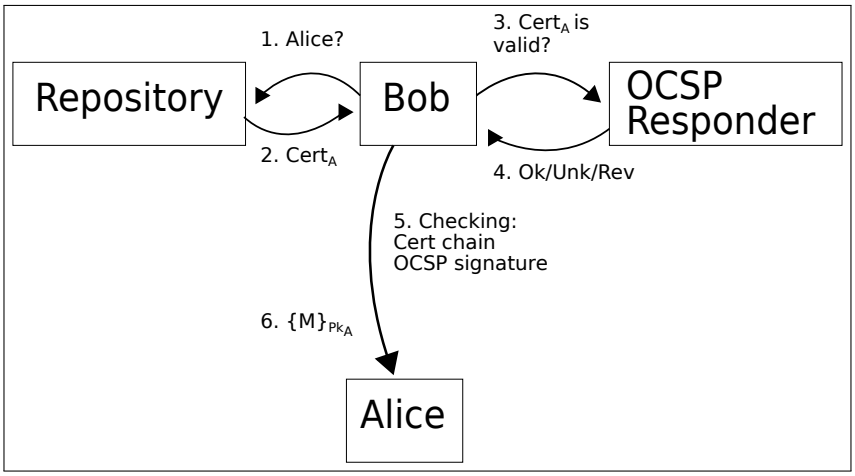

Fig. 4. *PKIX* end entity interactive authentication [11, Fig. 4].

3 Protocol Description

We now give some notations and then formally define the LOCALPKI protocols.

3.1 Notations

The concatenation of two messages m_1 and m_2 is denoted $m_1||m_2$. Generally, the notation O_A is used to express the belonging of the object O (e.g., a certificate) to the user A. We denote by Pk_A (resp. Sk_A) the public key (resp. private key) of a user A. We write $\{m\}_{Pk_A}$ (resp. $\{m\}_{Sk_A}$) the action of ciphering (resp. signing) a message m with the public key Pk_A (resp. the private key Sk_A). Hashing a message m is written $H(m)$, with H the hash function. We denote by $X_{509}()$ a function which takes user's information and returns them into the X_{509} certificate format without any signature.

3.2 Registration of a New User

A user first needs to be enrolled into the system: this step is called *registration*.

The phase starts by a new key pair generation by the user. After he interacts with a *LRA* for the registration process. In LOCALPKI the *LRA* should be physically close to the user, so that they can meet in person. The user begins by providing ID proofs according to the established security policy (e.g., a visual check of the ID card). Once the identity check succeeds, the user gives his public key to the *LRA*. Next, the authority generates a field equivalent to the *ToBeSigned* (TBSCert) in the $X.509$ certificates, containing at least: the user's ID, his public key, a *Serial Number* (*SN*), a validity period and the URL of the notary associated with the *LRA*. The *SN* has been obtained by the *LRA* from previous exchanges with the *EN*. The latter is in charge of the *SN* generation, and communicates to each supervised *LRA* a specific range of *SN*. For the next step, the

user hashes the previously generated $TBSCert$ and signs the digest: the result is called SI (*Signature Id*). Afterwards, by using the previously provided public key, the *LRA* is able to recompute the digest and then to check the correctness of the signature. At this step, the user has proven his knowledge about the associated private key to the *LRA*. The final registration phase is performed by the *EN* who registers the unique couple (SN, SI) (this is a simplified version of what is called a "public key ownership certificate" in [6]). For this, the *LRA* ciphers the couple using the *EN*'s public key, and sends the result along with its signature of this message (i.e., $\{H(SN, SI)\}_{Sk_{LRA}}$). In the end, the registered user owns a certificate $Cert$ containing the $TBSCert$, and in particular the SN and the SI, while the *EN* has added the associated couple (SN, SI) to his database. The complete process is detailed in Algorithm 1 and schemed in Figure 5.

Algorithm 1 Registration of Alice

Require: The *LRA* owns *a priori* serial numbers, provided by a trusted electronic notary *EN*.
Ensure: Identity check (by the *LRA*) and registration of Alice into the *EN* database.

1: Alice generates his public key Pk_A.
2: Alice $\rightarrow$ *LRA*: Pk_A
3: *LRA* checks information and identity of Alice.
4: *LRA* $\rightarrow$ Alice: Serial number (SN_A), notary URL (URL_{EN}), validity.
5: Alice generates a $X.509$ certificate $TBSCert_A$ (completed with URL_{EN} and SN_A), and computes $SI_A = \{H(TBSCert_A)\}_{Sk_A}$
6: Alice $\rightarrow$ *LRA*: $TBSCert_A || SI_A$
7: *LRA* checks SI_A (PoK of the Alice private key).
8: **if** Verification OK **then**
9: *LRA* $\rightarrow$ *EN*: $\{SN_A || SI_A\}_{Pk_{EN}} || \{H(SN_A || SI_A)\}_{Sk_{LRA}}$
10: **end if**
11: *EN* decides to add Alice to his database.

3.3 Authentication

Once the owner of the certificate has been correctly registered, other users could authenticate him i.e., they are ensured that the used public key is indeed the owner's one. The authentication process in LocalPKI can be realized in two different ways. First by using a private mode, where only the *EN* knows the full database containing registered users. In this case, the user requests the *EN* about the validity of the owner's certificate, and no more information is revealed. The second possibility is to apply the public mode (also called buffering mode), where the *EN* shares parts of his database with the user, who makes the validity verification by himself. In both cases, the user must be able to first authenticate the *EN* using the URL provided in the certificate. As in *PKIX*, the mechanism used in LocalPKI is the *trust anchors* [24]. A trust anchor contains certificates of the trusted notaries. Then, users are able to retrieve information on notaries, in particular their associated public key. In the following, we detail both the private and public modes.

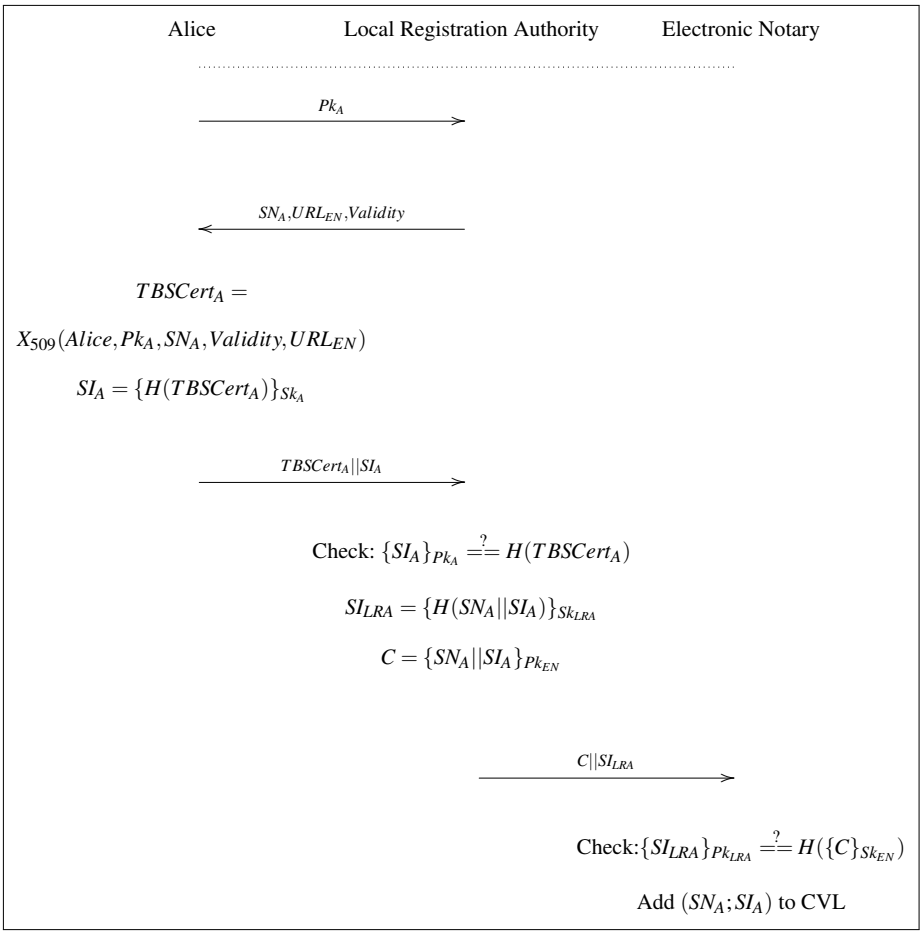

Fig. 5. Registration of Alice [11, Fig. 5].

Private Mode. The specificity of the private mode resides in the database of registered users, which is only known by the *EN*. Then the validity check of the owner's certificate by another user (the *verifier*) is made by interacting with the *EN* at verification time. First the verifier gets the certificate from the owner. Then, the verifier has two possibilities: he can check the owner's self signature by himself (Algorithm 2) or delegate also this task to the notary (Algorithm 3).

In the first case, the *Authentication Request* (*AR*) only contains the couple (SN, SI) and a nonce R. In the second case, the AR is made of the complete certificate and a nonce. In any case, the entity verifying the signature has to extract the public key, the *SI* and the $TBSCert$ from the certificate. Then, he hashes the $TBSCert$ and applies the cipher on the *SI* using the public key. The verifier also checks that the *EN* associated to the URL contained in the certificate belongs to his trust anchor, otherwise authentication cannot be realized. Then, he sends the correct AR (depending on the choice of the

Algorithm 2 Certificate check in private mode (self signature verification)

Require: Alice gets the certificate from Bob. She wants to check the validity of the certificate.
Ensure: Authentication of Bob to Alice if the certificate is correct, failure otherwise.

if $\{SI_B\}_{Pk_B} \stackrel{?}{=} H(TBSCert_B)$ **then**
  Alice: $R_A \leftarrow^{\$}$
  Alice $\rightarrow URL_{EN}$: AR=$SN_{Bob}||SI_{Bob}||R_A$
  if $(SN_{Bob}; SI_{Bob}) \in$ Database **then**
    $Rep =$ "OK"$||AR$
  else
    $Rep =$ "$Unknown$"$||AR$
  end if
  $URL_{EN} \rightarrow$ Alice: $Rep||\{H(Rep)\}_{Sk_{EN}}$
  Alice checks the response signature, and authenticate (or not) Bob.
end if

Algorithm 3 Certificate check in private mode (delegate signature verification)

Require: Alice gets the certificate from Bob. She wants to check the validity of the certificate.
Ensure: Authentication of Bob to Alice if the certificate is correct, failure otherwise.

Alice: $R_A \leftarrow^{\$}$
Alice $\rightarrow URL_{EN}$: AR=$Cert_{Bob}||R_A$
if $\{SI_B\}_{Pk_B} \stackrel{?}{=} H(TBSCert_B)$ **then**
  if $(SN_{Bob}; SI_{Bob}) \in$ Database **then**
    $Rep =$ "OK"$||AR$
  else
    $Rep =$ "$Unknown$"$||AR$
  end if
else
  $Rep =$ "$Wrong\ Signature$"$||AR$
end if
$URL_{EN} \rightarrow$ Alice: $Rep||\{H(Rep)\}_{Sk_{EN}}$
Alice checks the response signature, and authenticate (or not) Bob.

signature verification) to the indicated *EN*. During the next step, the *EN* looks for the given IDs in his database. If they are found, he responds positively, otherwise he gives a negative answer. The complete message consists of the previous answer, the nonce R, the couple (SN, SI) and the signature by the notary of all the previous contents. Finally, the verifier checks the *EN*'s signature of the answer using the public key extracted from the *EN*'s certificate (stored in the user trust anchor). An example of authentication can be found in Figure 6.

Public Mode: Certificate Verification List. In the public mode, checking the owner's certificate validity is realized by the verifier. He first obtains the owner's certificate. Then he must check that the certificate's signature (i.e., the *SI*) is consistent with the information contained in the *TBSCert* (i.e., the public key and the *SN*). After having checked that the *EN* belongs to his trust anchor, the verifier sends a request to the *EN*

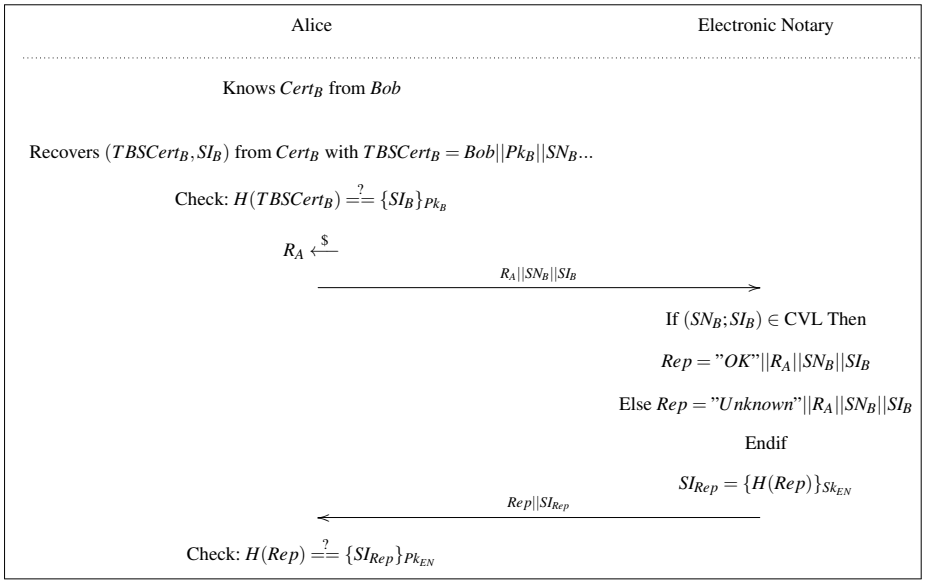

Fig. 6. Authentication of Bob by Alice (in private mode) [11, Fig. 6].

designated by its URL in the certificate. This request asks for the *Certificate Verification List* (*CVL*) i.e., the content of the database storing the couple (SN, SI) of previously registered users. To exchange the *CVL* with the user, the *EN* sends a signature of the *CVL* in addition to the list itself. This also ensures its integrity. Finally, the verifier checks the signature of the *EN*, and then verifies that (SN, SI) belongs to the *CVL*. Of course, just like a *CRL*, a *CVL* can be locally stored (buffered) and reused in a certain time interval without any refreshment. Just like for a *CRL*, a typical refreshment rate for a *CVL* could be in days. An example of public authentication is given in Figure 7.

Even if the public mode reduces the number of operation realized by the *EN*, it implies a non-negligible communication cost. In order to reduce it, the idea is to divide the database into subdomains. Indeed, the *EN* is in charge of the generation of the *SN* given to the *LRA*. Hence, the database is intrinsically divided into *LRA* subdomains. Then, the verifier could buffer only the *CVL* associated to a subdomain, and hence the communication cost between the client and the *EN* is reduced to this subdomain size.

Comparison between Public, Private Modes and *PKIX* Mechanisms. LOCALPKI is designed to be used by default in the private mode, offering a lower communication cost and an always up-to-date database. Nevertheless, the *CVL* are interesting in the case where an online interaction is not always guaranteed. In comparison with mechanisms used in *PKIX*, the private mode can be assimilated with *OCSP* and *CVL* with *CRL*. Initially, *OCSP* was not designed to resist against replay attacks. In a later version, a counter-measure has been added to the `responseExtensions` field (see [26, § 4.4.1]). However, as discussed in [26, § 5], this solution is not required by the norm. In the private mode of LOCALPKI, replay attacks are countered by default.

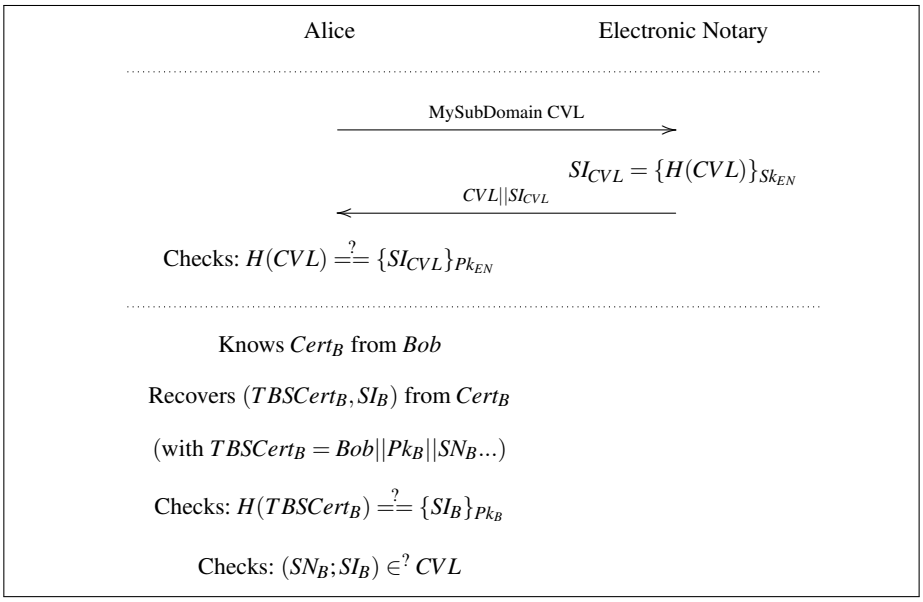

Fig. 7. Authentication of Bob by Alice (in public mode) [11, Fig; 7].

In *PKIX*, the complete list of revoked certificates must be shared. Thus, the communication cost is large. A way to reduce these large broadcasts, on solution can be to use δ-*CRL*'s. Similarly we proposed to allow subdomain CVL's. In the case of the *CVL*, only the subdomain part containing the certificates need to be exchanged to have a correct authentication. But, unlike δ-*CRL*, a subdomain *CVL* is fail-safe. Indeed a user looking only in a partial *CRL* can miss that a certificate has been revoked with another reason, and still use it. On the contrary, if a valid certificate is not present in a subdomain *CVL*, it just cannot be used.

Furthermore, *CRL* based solutions expose users to false positive authentications. Since a *CRL* is not continuously updated, recently revoked certificates could still be considered valid. By using a white-list strategy like *CVL*, verifiers may not succeed in authenticating newly registered users. However, once again, obtaining a false negative is usually a safer fail than a false positive. Besides that, only valid certificates are stored. Then, in case of authentication failure, the verifier cannot know the reason (revocation, unregistered user).

3.4 Revocation

A public key infrastructure should provide a solution to revoke certificates before the end of their validity period. This allows for instance to manage lost or compromised key pairs. In LOCALPKI, a certificate revocation can easily be done by the certificate's owner or by the *LRA*, via a request to the *EN*. In both cases, this request is signed and contains the owner's certificate along with a `Revoke` message, as shown in Algorithm 4 thereafter. In case of a request from the owner's certificate, the signature acts as a proof

of knowledge of the private key. In case of a request by the associated *LRA*, the notary has anyway to accept registrations from this *LRA* (for instance within a range of allowed serial number values). This same range can be used to guaranty that the *LRA* is allowed to revoke this serial number. Hence, in both cases, the *EN* verifies the signature and if the couple (SN, SI) is present in his database, he simply removes the entry. Thus, an authentication attempt using this revoked couple fails, since the entry has been removed from the *EN* database. Then the renewal procedure is simple: after the revocation, the user has to enter a new registration process with his *LRA*.

Algorithm 4 Certificate revocation

Require: Alice correctly registered in the *EN* database. $X \in \{LRA, User\}$
Ensure: The certificate revocation of User
1: $X \rightarrow EN$: $SI_{Rev} = Cert_X || \{H(\text{"}Revoke\text{"} || (SN_{User}; SI_{User}))\}_{Sk_X}$
2: *EN* checks signature
3: **if** Verification is OK and $(SN_X; SI_X) \in$ DataBase **then**
4: *EN* removes $(SN_X; SI_X)$ from the database.
5: **end if**

4 Infrastructures interoperability

4.1 Cross-certification tag

In *PKIX*, cross-certification is a mechanism allowing the interoperability between private PKI. This means that users from a PKI A, are then able to authenticate other users belonging to the PKI B. In practice, the cross-certification consists in the signature of a CA's certificate by another CA. In the end, this creates a trust path between the CAs, which is verified during the authentication process. The cross-certification is generally transparent for the end-users, since the technical checks are realized by the CA. In LocalPKI, we explain in the following how to define a similar and efficient interoperability system using private blockchain based techniques.

First, each notary should carefully look into the operational mode of the others, to be sure that their security policies do coincide. This is in general realized via a set of audits and agreements. Then, the notaries should tag each entry of their database with the issuing notary name. This process is needed to ensure the uniqueness of each database entry. In practice, the set of *SN* is individually managed by each *EN* and thus, inter-notary collisions might be possible. The tag could be the notary names, or a specific identifier which identifies them. A database entry of an *EN* with a tag TAG_k for a end-user indexed with i thus has the following format in a cross-certification setting:

$$TAG_k, SN_i, SI_i$$

4.2 Database exchange

A first solution ensuring the interoperability between several *EN*s could be to setup a databases exchange. For instance, the notaries EN_1 and EN_2 agree on a secure exchange protocol, where for instance mutual authentication could be performed out of band. Then, they simply exchange their own signed database, like in the *CVL* mechanism. In the end, both notaries obtain a shared database. This process could be sufficient when the PKI are locally deployed, e.g, when a company wishes to authenticate users belonging to two distinct subsidiaries, running two instances of LOCALPKI.

This is however inefficient in most cases: indeed, since revoked certificates must be removed from the database and new users can be frequently added, the whole databases would have to be frequently checked and exchanged. We propose an update mechanism that should reduce the communications volume and give the possibility to easily add and remove data from a shared database.

4.3 Private blockchain solution

In fact, the problem is quite exactly that of the management of accounts and their balance in the shared state of the *Ethereum* cryptocurrency [7]. There, the solution is based on the use of a structure called a *Merkle PATRICIA Tree* (MPT) [15]. This structure combines that of radix trees [21] with Merkle hash trees [20]. The first ones allow efficient additions or deletions of nodes, whereas the second use the nodes hashes as pointers to the next nodes. In the context of LOCALPKI, the leafs represent an entry of the *EN* database. Then, to update their mutual databases (which became a shared database thanks to the cross-certification), the notaries have only to update their MPT.

Overall the solution is thus to setup the database as a **private blockchain**, between the notaries involved in the cross certification, using Merkle PATRICIA trees as the underlying data structure.

5 Deployment

LOCALPKI has the advantage to be easily deployed from an existing *PKIX*. Actually, each of these requirements can be satisfied by adapting the current standards of *PKIX*. In the following, we present how to realize it.

Certificate format. First of all, certificates employed in LOCALPKI can be based on *X.509v3* certificates [9]. Indeed, both PKI, LOCALPKI and *PKIX*, share the same kind of identification data (TBSCert). Then, the *CA*'s signature is replaced by the user's signature and the *SN* can be stored in the `serialNumber` field.

On-line authentication request. The communications with the notaries during authentication in the private mode, Figure 6, can be set up using the *OCSP* norm [26]. Within the *OCSP* request, the *SN* replaces the current `serialNumber` in the `CertID` sequence, and the *SI* is stored in the `signature` field of the optional `Signature` sequence. Note that other fields in `CertID`, such that the `issuerNameHash` and `issuerKeyHash`, could be left empty since this information is either irrelevant or redundant with the *SI*. The nonce R_A (see Figure 6 and Algorithms 2 and 3) can be stored into the *OCSP* `requestExtensions`.

On-line authentication response. Similarly, the notary's answer can also follow the *OCSP* response format, where all the latter fields are also present.

Certificate Validation Lists. In the public mode, Certificate Verification Lists (*CVL*) can be managed just like Certificate Revocation Lists (*CRL*). For example, a *CVL* could be published on the *EN* websites, and can be stored in local repositories. Moreover, if an organization with subdomains is required, e.g., each range of *SN* represents a subdomain, the *Delta-CRL* indicator could used [9], as well as the `BaseCRLNumber` field which could represent for us an equivalent Base CVL number field.

Revocation. The revocation requests within LOCALPKI and *PKIX* are almost identical. Thus, LOCALPKI may use the *Certificate Management Protocol* [23] directly for revocations.

End-user authentication. All authentication mechanisms using X.509 certificates, like *Simple Authentication and Security Layer* (SASL) (e.g., via ISO IEC 9798-3), are still enabled with LOCALPKI.

Web of trust. Similarly, enhancements to the web of trust between authorities like ARPKI [4] or Certificate Transparency [18], which use append only log servers are also applicable to LOCALPKI.

Trust anchor stores. Finally, *PKIX* requires trust anchors [24] to be deployed, e.g., within the store of the users' browsers or OS's. Communications between *LRA*s and notaries, also require an anchor mechanism and that of *PKIX* can also be used directly.

Therefore, existing tools like *OpenSSL* allow for all entities to generate keys, certificates, authentication and revocation requests or responses. The technical setup of LOCALPKI is mainly restricted to the management of the databases. This is delegated to the *EN*, who are thus also in charge of maintaining the *PKI* availability. *LRA*s are in charge of communications with the notaries, that is mainly exchanging serial numbers, and of the face to face identity verifications.

Differently, users have several possibilities, depending on their expertise.

- Expert users, first generate themselves their own key pair. Then they request a serial number, *SN*, through the *LRA*, in order to create and sign their certificate. Finally they give the associated *SI* to the *LRA* who will forward it to the notary.
- An intermediate possibility, is for the user to only generate a key pair. The *LRA* will then take charge of the certificate creation and provide means for the user to sign it with his private key (for instance a usb port and a keyboard so as to type the password deciphering the private key stored in a usb device).
- The *LRA* can also create fresh key pairs on the fly and provide everything to the user.

Part or all of the latter two possibilities are easily realized through a dedicated web site. Therefore, the only technical requirement for the *LRA* is the use of tools like *OpenSSL*, in order to help the users registration and the creation of a user-friendly associated API.

We have realized a web-base prototype implementation of the system, available there:

`http://hpac.imag.fr/localpki`

This first prototype uses PHP to call some OpenSSL scripts as shown with the capture of Figure 8.

Fig. 8. Web-based prototype implementation of LOCALPKI

The main OpenSSL scripts do implement LOCALPKI functions as follows:

- As a LOCALPKI certificate is self signed, `openssl req -new -x509` can be used to generate it (for *PKIX*, it would b a self signed *root CA certificate*).
- To be handled by most web browser, the generated PEM format certificate can then be converted to the PKCS12 format, via `openssl pkcs12`.
- Additionally, `openssl genrsa` can be used to generate the public/private key pair for the user.

6 Security Analysis

Using asymmetric cryptography, LOCALPKI aims at making authentication of users possible. However, the protocol also provides other security guarantees. In the following, we first define the security properties of LOCALPKI. Then, we use an automatic cryptographic protocols verification tool called *Tamarin Prover*, in order to prove these security properties.

6.1 Security Properties

Firstly the protocol must be *correct*, i.e., if a person has been correctly registered into the database and her certificate is still valid, then he must be correctly authenticated. The underlying property correspond to classical authentication property and is thus about correct identity checks.

Vice versa, an adversary who has not been registered cannot be authenticated. This *soundness property* implies that an adversary cannot forge a certificate considered as valid and cannot impersonate a valid one. A reformulation of the soundness property is that authentication at time i_2 implies a registration at time i_1 with $i_1 < i_2$.

In [4], the authors also defined the *Connection Integrity* as follows: if a user establishes a connection with another one, then the user communicates with the legitimate owner of the private key. In others words, in the case where registration has been correctly done (i.e., by honest participants), no adversary can possibly know the private key of the honest owner.

Moreover, the protocol must ensure some secrecy properties. Indeed, a protocol execution must not reveal any sensitive information: once authenticated, the adversary cannot know the message.

To summarize, by assuming that *LRA* and the *EN* are trusted, LOCALPKI verifies the following security properties:

- *Correctness*: if a user has been correctly registered and is not revoked then he must be correctly authenticated ;
- *Soundness*: if a user has been authenticated, then he must have been registered before and he has not been revoked before;
- *Connection Integrity*: if a user is correctly registered, then the adversary does not know his private key.
- *Secrecy*: once a user is authenticated, the messages sent to him cannot be learnt by the adversary.

6.2 Tamarin Prover Modeling

We use *Tamarin Prover* an automatic security protocol verifier [19] to prove these security properties. This tool can verify an unbounded number of sessions. Moreover the intruder follows the Dolev-Yao's intruder model [10]. This means that the intruder extracts all possible information from every exchanged messages on the network. The tool assumes the usual perfect encryption hypothesis (i.e., the adversary cannot learn any information from encrypted messages if he does not know the corresponding secret key). We consider classic equational theories provided by default by Tamarin for the cryptographic operations like encryption or signature. In Tamarin, the protocols are modeled by some multiset rewriting rules. These rules are composed of facts, which model the local knowledge of a participant, such as the reception of a message, or the generation of a fresh number or the emission of a message. Then, each role action is implemented as a `rule`. A rule rewrites a fact into another one, and is eventually labeled in order to trace the realized actions. For example, a rule rewriting a fact composed of a message and key into a message containing the encryption of the message with the key could be

labeled as *Cipher*. Facts could be persistent (denoted with a `!` before its name), which means that it can be reused arbitrarily often by some rules. Otherwise facts are said to be linear and can be used only once. Once the protocol is modeled by some rules, we model the desired properties as `lemmas`. A lemma is a first-order logic sequence applied on the label of previously defined rules. They contain quantifiers (∀: `All`, ∃: `Ex`) and logical connectives (`&`, `|`, `not`, `==>`), along with timepoints (declared with `#`, and employed with `@`). For example `Ex sna sia #i. Bob_Auth_Alice(<sna,sia>) @i` means that the event *Bob Authenticates Alice* using variables `<sna,sia>` occurs at time `i`.

Now, we detail our model and the properties proved. We also show that trust hypothesis on entities are mandatory, just like, e.g., trust hypothesis are required on CAs in *PKIX*, by describing attacks found by the tool if any of these hypothesis is removed.

All the Tamarin source files can be found at: `http://hpac.imag.fr/localpki`. See the associated *Makefile* for details on how to generate proofs and attacks with Tamarin.

6.3 LocalPKI Tamarin Model

According to the result of [8], where the authors prove that for verifying a secrecy property only one intruder is enough and for an authentication property one intruder and at most one honest participant per role is enough, we consider a protocol execution where we have only one notary, one *LRA*, one registered user (Alice) and one verifier (Bob). Our model of this execution comprises the registration of Alice, her authentication by Bob followed by an exchange of a message, and the revocation of her certificate.

In *PKIX* paper, key pair generations are modeled by generic persistent facts, instantiated with different terms. These facts bind the identity of the entity with its public key to represent the trust anchors, i.e., other entities have the correct association between the public key and the identity. Thus, we model the trust anchors of the *LRA* and the *EN* in the same way. In the case of Alice, we need to explicitly define the self-signed certificate described in the Algorithm 1. Moreover, we assume also that the Alice's certificate is public. We have two types of communication channels, depending on the situation:

- a real-life meeting,
- insecure channels.

Since the registration must be realized during a real-life meeting, we assume that it is not subject to any intruder attack. Therefore, our model expresses the information exchanged between the *LRA* and Alice with a private channel (i.e., the adversary is not able to learn or modify any information for this exchange). All others communications are public and then controlled by the intruder and exposed to eventual wiretapping.

The registration of the couple (SN_A, SI_A) in the database is modeled as a state which associate the knowledge of (SN_A, SI_A) to the identity of the *EN*.

Finally, for revocation we have to represent the removal of the certificate from the database. For this we used a tag into a persistent fact to express that the revocation is definitive.

6.4 Security Properties

First, we explicit security properties i.e., lemmas, in the case where all entities are supposed honest but in presence of an intruder. As defined in the Section 6.1, we have implemented lemmas about correctness, soundness and secrecy of the protocol.

Soundness. The first lemma (`soundness`) proves the soundness of the LOCALPKI. The couple `<sna,sia>` represents *SN* the *Serial Number* and *SI* the *Signature Id* of Alice. Each label in the lemma represents when the actual event occurs. The idea is to prove that for all possible couples `<sna,sia>` related to the Alice's private key `ltkA`, if Bob has successfully authenticated Alice at time `i` (`Bob_Auth_Alice()`), then it means that Alice has been previously registered (at time `j`, by the notary, `EN_Reg_Alice()`), and that if the certificate has been revoked (`Cert_Is_Revoked()`), then it was at an earlier time `k`:

```
lemma soundness:
  all-traces
 "All EN B sna sia #i.
  Bob_Auth_Alice(B, <sna,sia>, ltkA) @i
  ==>
 (Ex #j. EN_Reg_Alice(EN, <sna,sia>, ltkA)@j
  & (j<i)) &
 (All #k.
  Cert_Is_Revoked(EN, <sna,sia>, ltkA)@k ==> i<k)"
```

Correctness. The second lemma (`correctness`) ensures that our model is correct, i.e., there exists an execution where Alice is registered, Bob authenticates Alice and the certificate is not revoked. Thanks to this lemma, we check that our model realizes the protocol steps in a correct order. Other lemmas in our modeling also provide sanity checks, for example to show that there exists a trace where the certificate has been correctly revoked.

```
lemma correctness:
  exists-trace
 "(Ex EN B sna sia ltkA #i #j.
  EN_Reg_Alice_(EN, <sna,sia>, ltkA) @i
  & Bob_Auth_Alice(B,<sna,sia>, ltkA) @j
  & not(Ex EN #l.
  Cert_Is_Revoked(EN, <sna,sia>, ltkA) @l & i<l))"
```

Secrecy. In the secrecy lemma (`secrecy`), we ensure the secrecy of exchanged messages using a LOCALPKI based authentication. Then, we prove in Tamarin that the message (denoted `x`) should not be in the adversary knowledge, written as `K()` at any time.

```
lemma secrecy:
  all-traces
  "All x #i. Secret(x) @i ==> not(Ex #j. K(x)@j)"
```

Connection Integrity. Similarly, in the connection integrity lemma (`connection`), we ensure that the secret key of Alice, `ltkA`, should not be in the adversary knowledge neither in the case where Bob authenticates Alice from her certificate nor in the case where the certificate has been revoked.

```
lemma connection:
  all-traces
  "(All EN sna sia ltkA #i.
    Bob_Auth_Alice(B, <sna,sia>, ltkA)@i
    ==> not(Ex #l. K(ltkA)  @l))
  & (All B sna sia ltkA #i.
    Cert_Is_Revoked(EN, <sna,sia>, ltkA) @i
    ==> not(Ex #l. K(ltkA)  @l))"
```

In other words, whatever subsequent messages sent by any user, Alice's private key remains private.

6.5 Trust assumptions

Next, we show that our trust assumptions in both the *LRA* and the *EN* are mandatory in order to preserve the security of the protocol (as is the case in *PKIX*). To show this we present attacks on the protocol where one of the entity is malicious i.e. its private key is given to the adversary.

Trust in the EN. In the case where the notary is corrupted, the secrecy is not verified because he is able to add a false key pair to his database. Then after the authentication of Alice by Bob with these keys, a wiretap between communications allows him to retrieve secret exchanged information. Soundness is also falsified: since the adversary has access to the *EN*'s private key, he acts as Alice has been previously registered into the *EN* database whereas she is not. In practice, this attack represents the possibility to the notary of giving to the person of his choice the trust to be authenticated. Concerning the connection integrity property, Tamarin finds an attack as soon as the *EN*'s key is leaked to the adversary. The attack is the following: after its initialization, the *EN* registers himself as Alice. This means that the adversary forges a certificate using Alice's identity, and uses its own private key to sign the certificate. At the next step, Bob authenticates Alice using this certificate, and the adversary knows the associated private key: therefore the connection integrity is broken. First, this attacks highlights the obvious forgery ability of a malicious notary. Second, it shows that the connection integrity property is well-defined. Indeed, such a forgery ruins the integrity, even if the adversary has no knowledge of the Alice's actual private key. This proves that the *EN* should be a trusted entity.

Trust in the LRA. When the *LRA*'s private key is leaked, the tool does not find any attack on the soundness. From a practical point of view, this result makes sense: the *LRA* is only involved into the registration process. Then, even if he provides false information, Alice should be registered before an authentication process. However, by removing the restriction of a unique registration per couple (SN_A, SI_A), the adversary sends a revoked couple to the *EN*, in order to pass over the revocation. In practice, serials numbers are provided by the *EN* so that this attack is easily preventable, by marking the used ones. Secrecy is falsified because the adversary impersonates the *LRA* during the last step of registration, by signing false information (i.e., false key pair). Then, since Bob does not exchange messages with the correct Alice's public key, and a simple wiretap allows

the adversary to retrieve secret messages. When the *LRA*'s private key is given to the adversary, the connection integrity is also broken. During this attack, Alice registers herself to the *LRA*. Then the *LRA* modifies Alice's certificate by including its own key pairs: the public one into the certificate, and the private one is used to sign the forged certificate. Then, the fake certificate is sent to the *EN*. Afterwards, suppose that the *EN*, for any reason, decides to revoke this certificate. At this point in time, the *EN* has succeeded in revoking the certificate while the adversary knows the associated private key. Hence, the connection integrity is broken. This attack shows the *LRA* ability to forge certificates. In conclusion, this proves that *LRA* should be a trusted entity.

Trust in Alice. When Alice is corrupted, repercussions on the protocol security are limited. It does not influence soundness property. This result is coherent: even with Alice's private key is given to the adversary, he cannot be authenticated without a previous registering. In practice, a malicious behavior from Alice could be an identity fraud during the registering if the *LRA* has been misled. However, this kind of attacks are out of the Tamarin scope, since ID checking cannot actually be modeled. On secrecy aspects, obviously, if the private key of Alice is leaked, the adversary is able to read the secret messages sent to Alice. This attack could correspond to the Alice's private key theft, where both Alice and the attacker are able to retrieve messages ciphered with the associate public key. Finally, if the private key of Alice is leaked, connection integrity is obviously broken, since the adversary can then use her key. Overall, we conclude that Alice does not need to be a trusted participant.

6.6 Security of the LocalPKI

From our security analysis, using the Tamarin prover, we have proven the following theorem:

Theorem 1. *If the notary and the local registration authority are not corrupted, then the* LocalPKI *security architecture is correct, sound, and preserves confidentiality and integrity.*

Our implementation consists of about 350 source lines of code to describe the model consisting of 20 rules and 4 lemmas. In order to reduce proof timings we have associated each entity to a defined `Role`. Moreover, we have enforced some actions to be unique (denoted as `Only_One`). These constraints are called `axioms` into the code. They avoid generic rewriting of rules either by associating a fixed string to an identity variable or either by implying timepoints equality in case of multiple uses of the same rules. Therefore, using a single core of an *i5 4590@3.50Ghz* with *8GB RAM*, we obtain very good performance for the security verification, as shown in Table 1.

7 LocalPKI as an *Internet of Things* PKI

By design, LocalPKI is made to be deployed into services which are close to end-users. Moreover, this architecture also presents many advantages if used within an highly constrained environments like *Industrial Control Systems* or, more generally, the

Table 1. Timings of Tamarin's proofs of lemmas [11, Tab. 1].

Lemma	CPU Time
Correctness:	4.7 s
Soundness:	4.4 s
Connection integrity:	10.9 s
Secrecy:	6.6 s

Internet of Things (IoT). IoT is a paradigm where a wide variety of objects, like wireless sensors or mobile phone, are able to interact each others to accomplish a common objective [1][14]. Because of the diversity of the architecture components, reaching security, or at least data integrity and authentication, is arduous. Particularly, this means that no assumption can be made on the computational nor communicating capabilities of the equipment. Another strong constraint of an IoT infrastructure lies into its setup instability: many components are frequently added or removed, and thus the certification process becomes complicated. In the following, we show that using LocalPKI as a security architecture for the IoT is a solution to some of the previous constraints.

In IoT, we can define a hierarchy: the manufacturer of the objects, the customer, and finally the objects [2]. For instance, one can refer to medical equipment, where the manufacturer sells to the hospital the connected devices. Here, there is a natural analogy with the LocalPKI actors: the manufacturer is associated with the *EN*, the customer with the *LRA* whereas the *things* represent the end-users, as shown in Figure 9. Each device is the owner of its own LocalPKI certificate.

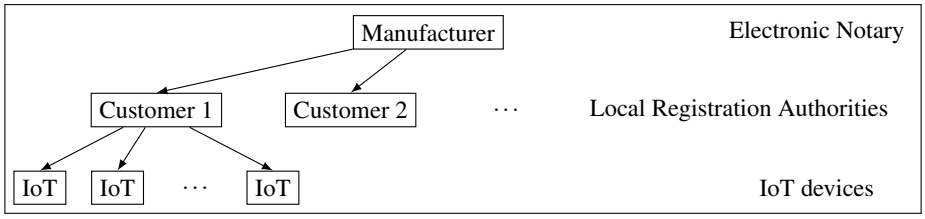

Fig. 9. LocalPKI as an *Internet of Things* PKI

The setup works as following: the manufacturer produces his products, and associates to each one of them a *Serial Number*. Then, he sells them to the customer. At this point, the latter is able to generate their certificate according to the protocol 1. Furthermore, the customer (*i.e.* the *LRA*) has the possibility to send by batch all the requests to complete the *EN*'s database. The manufacturer then acts as an Electronic Notary and stores the certificates database. The customers acting as local registration authorities and, potentially with the help of the manufacturer, configure the devices and authorize them onto the network. As already mentioned, then any two devices are able to perform mutual authentication with any challenge-response protocol, like SASL. Note that cus-

tomers also have the possibility to recover the database and set up their private notary system.

From a practical point of view, this whole scheme allows to generate the certificates out of bands, and then to communicate all the requests at the same time when a connection is available. The global communications volume is quite similar to *PKIX*, but the distribution is different. In LOCALPKI, the majority of the communication is local, *i.e.* it lies between the *LRA* and the end-users, whereas the registration authority of *PKIX* is generally farther. Indeed, before authentication and confidential communication processes, each device sends its public key certificate to each other. For both authentication and confidential communication processes, each device must be sure that the public key of the other device is authentic. To do so, each device can either use the public or private modes of Section 3.3.

During the registration process in LOCALPKI, when the end-users and the *LRA* agree on the certificate, the objects are ready to be deployed. On the contrary, with *PKIX* the objects should had to wait for the *Certification Authority* signature. This means that the objects cannot be fully deployed, since they do not have their certificate, until the authority responds to the requests. Here, with LOCALPKI, once the certificate has been created, it is stored into the object, and no more communication with is required between the object and the *EN*. Then, the objects could be immediately distributed. Of course, just like *PKIX*, secure communications are not possible until the *EN* validates the *LRA* requests. The main advantage of LOCALPKI there is thus to be asynchronous.

Furthermore, LOCALPKI provides a more suitable fail-safe system than *PKIX* in IOT. This comes from the use of white lists, which contains the valid certificates. On the one hand, an update is needed to allow secure communications with new objects which, as previously said, could be done by batches. On the other hand, these lists offer a fine-grained controls on the set of authorized connected objects by giving them access on the secure network only after they have been registered.

8 Conclusion

In this paper, we have proposed an alternative public-key infrastructure model, LOCALPKI. This model has been analyzed and its security formally proven using Tamarin. The main feature is that, unlike the *PKIX* standard, here user certificates are self-signed and only the binding of a certificate with an identity is signed by a third-party, a notary in LOCALPKI. Therefore, it is easier for users to use certificates within their local environment. For instance, the registration process can be transferred to the notaries by local businesses, which only have to handle identity verifications. We think that this could foster a wider spread of certificates among everyday end users. For this, notaries just have to maintain an accessible database of fingerprints. Hence, LOCALPKI is an alternative to the *PKIX* solution, providing similar security properties.

Furthermore, we have shown in Section 7 that this is particularly well suited to constrained user-environments, such as Industrial Control Systems or, more generally, the Internet of Things. In particular, the asynchronous certification process allows to have an easy and flexible setup. Also, in some cases, the key management of LOCALPKI is better than that of *PKIX*. Another example is for local businesses as a bank, where

bank customers have the possibility to look at their account from a website, quite often using a certificate-based authentication. Managing a classical *PKIX* either requires a large internal department or a full delegation of trust to a *PKI* actor. By deploying the LOCALPKI solution, the bank still preserves a local registration, and thus trust and responsibility of its customers, and the technical part of ensuring their authentication is deported to notaries. Overall, the cost and the technical requirements for the company are reduced. Moreover, LOCALPKI also offers to small agencies the possibility to help users for their registration. The economics incentive would be to offer this service only at a moderate cost, where deploying a full *PKI* in every agency should be too expensive.

The main similarities and differences between *PKIX* and LOCALPKI are shown in Table 2:

Table 2. Comparing LOCALPKI and *PKIX*

	LOCALPKI	*PKIX*
Certificate creation	User & LRA	CA
Certificate signature	User	CA
Certificate registration	EN	/
Default authentication	Interactive (*Private mode*)	Off-line (*CRLs*)
Alternative authentication	Off-line (*Public mode, CVLs*)	Interactive (*OCSP*)
Formal trust	LRA & EN	RA & AC

Finally, as seen in Section 5, the overall deployment of LOCALPKI can be made using only existing tools and formats. This facilitates the process, as is demonstrated with our PHP and OpenSSL prototype available implementation: `http://hpac.imag.fr/localpki`.

Further work include:

- legal aspect on the shared responsibility of the *LRA* and the *EN* in LOCALPKI should be studied. Indeed, both entities play an important role into the registration and authentication processes. Our first idea is that the *EN* could register the identity of the *LRA* sending information about new users and store this information also within the database. Thus, if this user is recognized as malicious, the *EN* could blame the *LRA*.
- trust enhancement of the authorities, just like for *PKIX*, is also an open problem. Adapting the secure trust computations described in [12] might be easier for LOCALPKI though. There, they compute a global trust by using the trust evaluation of each certification authorities towards the others. This allows to obtain a better hint about the trust given in *CAs*, instead of being based only on the trustfulness given by trust anchors. This method could be relevant to the LOCALPKI architecture, since it is also using the trust anchor mechanism.

– whether this model could also simplify identity-based approaches like certificate-less PKI [13,3], for instance using attributes.
– Finally, just like *Let's Encrypt* offers certificates enabling `HTTPS` (via SSL/TLS) for websites, we have set up a web site offering the possibility to create self-signed LocalPKI certificates for any user and connected device; we plan to enhance the capabilities of this web-site to manage our other protocols, such as revocation, cross-certification, on-line authentication, database paging, etc.

Acknowledgment

We thank Amaury Huot for his help in implementing the prototype web-based interface to LocalPKI certificates.

References

1. L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805, 2010. URL: http://www.sciencedirect.com/science/article/pii/S1389128610001568, doi:https://doi.org/10.1016/j.comnet.2010.05.010.
2. B. Badrignans, V. Danjean, J.-G. Dumas, P. Elbaz-Vincent, S. Machenaud, J.-B. Orfila, F. Pebay-Peyroula, F. cois Pebay-Peyroula, M.-L. Potet, M. Puys, J.-L. Richier, and J.-L. Roch. Security architecture for point-to-point splitting protocols. In *IEEE World Congress on Industrial Control Systems Security, Cambridge, UK*, page 8, Dec. 2017. URL: https://hal.archives-ouvertes.fr/hal-01657605.
3. J. Baek, R. Safavi-Naini, and W. Susilo. Certificateless public key encryption without pairing. In *Information Security, 8th International Conference, ISC 2005, Singapore, September 20-23, 2005, Proceedings*, volume 3650 of *Lecture Notes in Computer Science*, pages 134–148. Springer, 2005. URL: http://dx.doi.org/10.1007/11556992_10, doi:10.1007/11556992_10.
4. D. Basin, C. Cremers, T. H.-J. Kim, A. Perrig, R. Sasse, and P. Szalachowski. ARPKI: Attack resilient public-key infrastructure. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 382–393, November 2014. URL: http://dx.doi.org/10.1145/2660267.2660298, doi:10.1145/2660267.2660298.
5. J. Bau and J. C. Mitchell. A security evaluation of DNSSEC with NSEC3. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2010, San Diego, California, USA, 28th February - 3rd March 2010*. The Internet Society, 2010. URL: http://www.isoc.org/isoc/conferences/ndss/10/pdf/17.pdf.
6. S. Bouzefrane, K. Garri, and P. Thoniel. A user-centric PKI based-protocol to manage FC2 digital identities. *IJCSI International Journal of Computer Science Issues*, 8(1):74–80, Jan. 2011. URL: https://hal.archives-ouvertes.fr/hal-00628633.
7. V. Buterin et al. Ethereum white paper, 2013.
8. H. Comon-Lundh and V. Cortier. Security properties: two agents are sufficient. *Sci. Comput. Program.*, 50(1-3):51–71, 2004. URL: http://dx.doi.org/10.1016/j.scico.2003.12.002, doi:10.1016/j.scico.2003.12.002.
9. D. Cooper. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, May 2008. URL: https://rfc-editor.org/rfc/rfc5280.txt, doi:10.17487/rfc5280.

10. D. Dolev and A. C. Yao. On the security of public key protocols. In *Proceedings of the 22Nd Annual Symposium on Foundations of Computer Science*, SFCS '81, pages 350–357, Washington, DC, USA, 1981. IEEE Computer Society. URL: http://dx.doi.org/10.1109/SFCS.1981.32, doi:10.1109/SFCS.1981.32.
11. J.-G. Dumas, P. Lafourcade, F. Melemedjian, J.-B. Orfila, and P. Thoniel. Localpki: A user-centric formally proven alternative to pkix. In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications - Volume 6: SECRYPT, (ICETE 2017)*, pages 187–199. INSTICC, SciTePress, 2017. doi:10.5220/0006461101870199.
12. J.-G. Dumas, P. Lafourcade, J.-B. Orfila, and M. Puys. Private multi-party matrix multiplication and trust computations. In *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016)*, pages 61–72, 2016. doi:10.5220/0005957200610072.
13. C. Gentry. Certificate-based encryption and the certificate revocation problem. In *Proceedings of the 22Nd International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'03, pages 272–293, Berlin, Heidelberg, 2003. Springer-Verlag. URL: http://dl.acm.org/citation.cfm?id=1766171.1766194.
14. D. Giusto, A. Iera, G. Morabito, and L. Atzori. *The Internet of Things: 20th Tyrrhenian Workshop on Digital Communications*. Springer Publishing Company, Incorporated, 2014.
15. W. E. Hall and C. S. Jutla. Parallelizable authentication trees. In B. Preneel and S. Tavares, editors, *Selected Areas in Cryptography*, pages 95–109, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
16. T. H.-J. Kim, L.-S. Huang, A. Perrig, C. Jackson, and V. Gligor. Accountable key infrastructure (AKI): A proposal for a public-key validation infrastructure. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 679–690, New York, NY, USA, 2013. ACM. URL: http://doi.acm.org/10.1145/2488388.2488448, doi:10.1145/2488388.2488448.
17. O. M. Kolkman, M. Mekking, and R. M. Gieben. DNSSEC Operational Practices, Version 2. RFC 6781, Dec. 2012. URL: https://rfc-editor.org/rfc/rfc6781.txt, doi:10.17487/rfc6781.
18. B. Laurie, A. Langley, and E. Kasper. Certificate Transparency. RFC 6962, June 2013. URL: https://rfc-editor.org/rfc/rfc6962.txt, doi:10.17487/RFC6962.
19. S. Meier, B. Schmidt, C. Cremers, and D. A. Basin. The TAMARIN prover for the symbolic analysis of security protocols. In N. Sharygina and H. Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 696–701. Springer, 2013. URL: http://dx.doi.org/10.1007/978-3-642-39799-8_48, doi:10.1007/978-3-642-39799-8_48.
20. R. C. Merkle. A digital signature based on a conventional encryption function. In C. Pomerance, editor, *Advances in Cryptology — CRYPTO '87*, pages 369–378, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.
21. D. R. Morrison. PATRICIA – practical algorithm to retrieve information coded in alphanumeric. *J. ACM*, 15(4):514–534, Oct. 1968. URL: http://doi.acm.org/10.1145/321479.321481, doi:10.1145/321479.321481.
22. J. L. Muñoz, O. Esparza, J. Forné, and E. Pallares. H-ocsp: A protocol to reduce the processing burden in online certificate status validation. *Electronic Commerce Research*, 8(4):255, 2008. URL: http://dx.doi.org/10.1007/s10660-008-9024-y, doi:10.1007/s10660-008-9024-y.
23. M. Peylo and T. Kause. Internet X.509 Public Key Infrastructure – HTTP Transfer for the Certificate Management Protocol (CMP). RFC 6712, Sept. 2012. URL: https://rfc-editor.org/rfc/rfc6712.txt, doi:10.17487/rfc6712.

24. R. Reddy and C. Wallace. Trust anchor management requirements. RFC 6024, RFC Editor, October 2010. URL: https://rfc-editor.org/rfc/rfc6024.txt.
25. M. D. Ryan. Enhanced certificate transparency and end-to-end encrypted mail. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*. The Internet Society, 2014. URL: http://www.internetsociety.org/doc/enhanced-certificate-transparency-and-end-end-encrypted-mail.
26. S. Santesson, R. Ankney, M. Myers, A. Malpani, S. Galperin, and D. C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 6960, June 2013. URL: https://rfc-editor.org/rfc/rfc6960.txt, doi:10.17487/rfc6960.
27. B. Schmidt, S. Meier, C. J. F. Cremers, and D. A. Basin. Automated analysis of Diffie-Hellman protocols and advanced security properties. In S. Chong, editor, *25th IEEE Computer Security Foundations Symposium, CSF 2012, Cambridge, MA, USA, June 25-27, 2012*, pages 78–94. IEEE Computer Society, 2012. URL: http://dx.doi.org/10.1109/CSF.2012.25, doi:10.1109/CSF.2012.25.
28. J. Vcelak, S. Goldberg, and D. Papadopoulos. NSEC5, DNSSEC Authenticated Denial of Existence. Internet-Draft draft-vcelak-nsec5-03, Internet Engineering Task Force, Sept. 2016. Work in Progress. URL: https://tools.ietf.org/html/draft-vcelak-nsec5-03.
29. J. Yu, V. Cheval, and M. Ryan. DTKI: A new formalized PKI with verifiable trusted parties. *Comput. J.*, 59(11):1695–1713, 2016. URL: http://dx.doi.org/10.1093/comjnl/bxw039, doi:10.1093/comjnl/bxw039.
30. P. R. Zimmermann. *The Official PGP User's Guide*. MIT Press, Cambridge, MA, USA, 1995.