2015 IEEE 12th International Conference on e-Business Engineering

Attribute-Based Encryption goes X.509

Florian Reimair, Johannes Feichtner, Peter Teufl
Institute for Applied Information Processing and Communications
Graz University of Technology, Austria
{florian.reimair, johannes.feichtner, peter.teufl}@iaik.tugraz.at

***Abstract*—Key authentication as well as an intended recipient not having a key available are, among others, challenges that public key infrastructures (PKIs) still face. Trusted third parties work around these issues. However, identity-based encryption (IBE) systems and later attribute-based encryption (ABE) systems were designed to address these exact challenges. Unfortunately, such schemes became only practicable after public key infrastructures have been picked up by industry. In this work, we present our approach on standing the above mentioned challenges. We propose to utilize recent developments on centralized key storage solutions to bring the features of IBE/ABE systems to PKI-based IT infrastructures. We describe our IBE/ABE emulation approach, present our prototype and give a thorough security evaluation. We found that it is possible to emulate IBE/ABE schemes without compromising security.**

***Index Terms*—key authentication, identity-based encryption, attribute-based encryption, public key infrastructures**

I. Introduction

Key authentication has been a challenge for public key infrastructures (PKI) ever since [21]. I. e. how can a sender know if the key she chose is indeed the key of the intended recipient. Trusted third parties, acting as certification authorities (CAs), were introduced to tackle the issue in practice. CAs do validate an identity and, on success, certify a public key in the form of a certificate. A sender can now pick a certificate and see which CA has certified the public key to be the key of which entity. Thus, the sender has to trust on the integrity of the CA. Aside from the key authentication challenge, another challenge is imposed on a sender when the meant recipient does not yet exist or has no public key data available. A recipient in this case can be a single individual or a group of individuals.

There, however, is identity-based encryption (IBE). Introduced in 1985 by Shamir [19], IBE has been designed to address these exact challenges. However, only 15 years later, the first efficient schemes were elaborated [2][4][18]. After that, the concept of IBE became more-widely known and variations, such as hierarchical IBE [10], accountable IBE [1], traceable IBE [7] appeared among others. 20 years after the original IBE scheme was presented, Sahai et. al. created a derivation of the scheme which they entitled fuzzy IBE [17] – which is nowadays referred to as Attribute-based encryption (ABE). Again, researchers created variations to fix issues and meet new use cases. Among these are key-policy ABE [8], multi authority ABE [3], distributed ABE [13], and decentralized ABE [12]. As of today, IBE/ABE provides a rich cryptographic toolbox.

However, the need for cryptography rose before IBE/ABE encryption schemes became available and public key infrastructures (PKIs), though being a long way from perfect, were adopted by industry. PKI also matured over time [9] and in the end, IT mainly works with PKI and has been standardized as X.509 [5][24]. The above mentioned challenges do, however, still exist and IBE/ABE schemes would – at times – offer better solutions. Unfortunately, IBE/ABE is not compatible with X.509 and PKI processes and therefore cannot be used without major reorganisation efforts.

There have, to the best of our knowledge, no attempts to merge the advantages of PKI and IBE schemes yet. There are, however, some solutions and attempts that disclose its great potential. Industry-created cloud services which are capable of encrypting data with keys that are hosted by the service, but they generally have to employ non-standard APIs for their functionality. Austria's mobile eID solution has equipped every citizen with a key pair and, therefore, would solve the issue of recipients not having certificates available. Yet, the eID solution is strictly tailored to qualified signatures and authentication.

Within this work, we present our approach of bringing IBE/ABE concepts towards today's IT. We were able to import the basic players and communication lines as known from the general IBE/ABE concept but replaced the central private key generator with our own service. Our service utilizes identity and attribute providers as well as a certification authority and has to satisfy only two requirements: first, there has to be directory service feature available, where a sender can retrieve public key data for a given set of attributes. Second, if there is no key available that matches the given set of attributes, a new key has to be created on-demand.

We give a detailed description of our prototypical implementation as well. We used the (semi-)centralized key service provided by CrySIL [16] as a basic infrastructure and enhance it to feature the requirements mentioned above. The prototype was tailored to reflect an enterprise deployment scenario where we utilize include Microsoft's Active Directory service as an identity and attribute provider as well as a company-owned certification authority for key certification.

In order to understand the features, possibilities, and security implications, we performed a functional evaluation as well as a thorough security- and risk anaysis. The functional evaluation shows that we succeeded in emulating IBE/ABE concepts and brought them to PKI-driven IT. The security analysis and risk analysis shows that we suffer from the rogue central service as IBE/ABE does from its central service. In total, our evaluation shows that it is feasible to emulate IBE/ABE concepts and bring their advantages to PKI solutions

978-1-4673-8002-7/15 $31.00 © 2015 IEEE
DOI 10.1109/ICEBE.2015.73

without compromising the security of the concepts.

The remainder of this work is structured as follows. Section II lists some related work and sets in relation to our goals. Section III discusses the inner workings of IBE/ABE scheme to a dregree to understand our approach, which we present in Section IV. Section V presents our prototype, Section VI evaluates our approach in general. Section VII concludes the work.

II. Related Work

Today's common cryptographic demands are mainly met with standard public-key infrastructure (PKI) systems. However, they are either free of charge but hard to use (e.g. PGP [20][23]) or available at a charge and easier to use but not guaranteed to be without risk or failure [6][22]. Aside from that, the basic challenge of PKI systems is to remedy the key authentication problem [21], e.g. how can one can be sure that a chosen key/certificate indeed represents the intended identity?

Multiple PKI systems show great potential for a highly trusted and usable solution. The Austrian mobile phone signature solution [14], for example, while implementing Austria's Citizen Card concept [11], manages cryptographic keys for every citizen on a central but state-owned hardware security module (HSM). Consequently, every citizen already has a cryptographic key that can be addressed by others. Unfortunately, the concept is designed for qualified digital signatures only and does neither provide support for encryption, nor does it provide a key directory service.

Industry also created promising solutions. *SigningHub*[1], *Cryptomathic*[2], and *Dictao*[3], among others, offer cryptographic services to enterprises. All of them support digital signature services, some even offer encryption services. Cryptographic keys are generally stored centrally and key usage is restricted through authentication and authorization processes, be it simple username/password or stronger methods (eID cards, OTPs, mobile devices etc). Some of the services do offer smart card solutions as well. Some of them are deployed as a cloud service, others are deployed in enterprise IT infrastructures. Having centralized key storage and authentication processes available, is a good basis for easy-to-use cryptography, however, these industry solutions mostly provide their services through proprietary APIs or applications which in general are not compatible with standard client applications. All in all, they fail in bringing IBE/ABE concepts to standard applications but pursue interesting approaches.

III. IBE/ABE in a nutshell

Identity-based encryption (IBE) has been around for quite some time now. A more general approach, fuzzy IBE [17] – also known as attribute-based encryption (ABE) – was proposed later. IBE/ABE basically works as follows: a data sender (*Alice*) derives a cryptographic key from a set of attributes and with the help of some global public key, issued by a central entity commonly referred to as *private key generator* (PKG). Attributes can be an email address as for IBE or some set of email addresses, group identifiers, dates, and so on for the more general ABE. *Alice* now sends the resulting cipher text to the recipient *Bob*. *Bob* consults the PKG, proves to the PKG that he indeed possesses the required attributes and the PKG generates a private key for *Bob*. *Bob* can now use this very key to decrypt the cipher text he received from *Alice*.

Thereby, IBE/ABE has two distinct advantages over widespread PKI-inspired solutions. First, there is no need to authenticate a key, i.e. making sure, a key/certificate belongs to the indented recipient identity. The sender only derives a key from a set of attributes and the recipient has to authenticate accordingly. Of course, there remains room for error as one may mistype certain attributes. Nevertheless, there is no need to trust on the identity mapping as it is necessary for PKI solutions. Second, since there is a central service, the PKG, whose main task is to generate private keys for a certain set of attributes, a rogue PKG can eavesdrop on the keys and – consequently – on the encrypted data. This inherent key escrow issue, however, well fits the original military use case, that required for the option, that in case of an emergency, any data could be decrypted. [1] and [7], introducing accountable server IBE and traceable IBE, respectively, provide solutions to use cases where the inherent key escrow property is not desired.

A more formal description of the IBE/ABE scenario is given in this paragraph. The first player in the scenario is the sending entity *Alice*. *Alice* wants to encrypt data without having to trust some third party on key authenticity. The receiving entity, *Bob,* wants to receive encrypted data. As he can be among the recipients in various ways (e.g. by email address, by group, by date, etc.) he does not want to keep distinct private keys around to match any of these combinations. Finally, the central *PKG* service provides the sending entity *Alice* with a master public key mpk, and derives and provides private recipient keys rk_{priv} to the recipient after the recipient has proven possession of the required attributes. The players and their communication paths are illustrated in Figure 1.

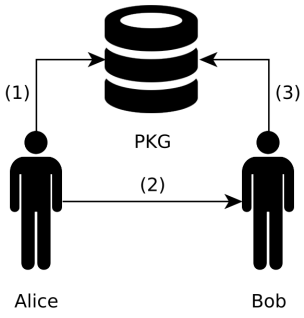

Figure 1. IBE/ABE illustration

The formal definitions of the steps of the IBE/ABE encryption scheme are given in Figure 2. The first step in an IBE/ABE process is to retrieve the public master key mk_{pub}

[1]http://www.signinghub.com
[2]http://www.cryptomathic.com
[3]https://www.dictao.com

via equation (1) (step (1) in Figure 1). After this initial step, *Alice* can proceed and encrypt arbitrary data by providing the recently acquired public master key mk_{pub}, a set of attributes $attrs$, and the plain payload data p to equation (2). *Alice* sends

$$getKey() = mk_{pub} \quad (1)$$
$$encrypt(mk_{pub}, attrs, p) = c \quad (2)$$
$$derive(mk_{pub}, mk_{priv}, attrs) = rk_{priv} \quad (3)$$
$$decrypt(mk_{pub}, rk_{priv}, c) = p \quad (4)$$

Figure 2. IBE/ABE definitions

the resulting cipher text c to the recipient *Bob* (step (2) in Figure 1). By now, *Bob* has the cipher text and can attempt to decrypt the data. In order to get the appropriate private key, the recipient's private key rk_{priv}, he has to provide the required set of attributes to the *PKG* service. The PKG service then provides its master key pair mk_{pub}, mk_{priv} and the received attributes $attrs$ to equation (3) and returns the resulting private recipient key rk_{priv} to *Bob*. *Bob* now has anything he needs to finally decrypt the data p he received from *Alice* by executing equation (4).

In total, this scheme does omit key authentication entirely by deriving a specific key for the envisioned group of recipients. The recipients themselves have to prove that they are part of the group of recipients. However, as the central *PKG* service can eavesdrop on the keys, derived by equation (3), the *PKG* may compromise security. IBE/ABE therefore shifts the issue of authenticating keys (practically solved through trusted entities such as CAs or webs-of-trust) to trusting the central *PKG*.

IV. Emulating IBE/ABE with X.509

For today's multi-device users, it is quite challenging to have their private key data securely transfered and stored on all devices, though the devices have different key protection capabilities. This brought new attention to the commonly known key distribution challenge and researchers and industry proposed centralized crypto services, among others, to meet the rather new requirements. The availability of such services is the basis of our approach on bringing IBE/ABE features into practice.

For our approach to work, however, we need the central crypto service (to which we will later refer to as central security module (SM)) to meet some special requirements: First, the service has to follow the most prominent crypto standard used in practice, namely X.509 [5][24]. X.509 fits nicely into existing IT infrastructures, such as the Secure/Multipurpose Internet Mail Extensions (S/MIME) for email applications and the Cryprographic Message Syntax (CMS) for data encryption needs. Second, our SM has to be trusted by the recipient *Bob*. That is, because *Bob*'s private key data is handled by the SM. Ideally, *Bob* owns the SM as a whole. Last but not least, the SM must create keys/certificates on demand.

Having a central SM enables us to emulate the IBE/ABE processes as follows: The players are similar as described in Section III: *Alice* wants to send encrypted data to *Bob* without having to do a key authentication. *Bob* wants to decrypt the data sent to him but does not want to keep various private keys at his device. Only the PKG player is replaced by our central SM. The players required by our approach and their communication paths are illustrated in Figure 3.

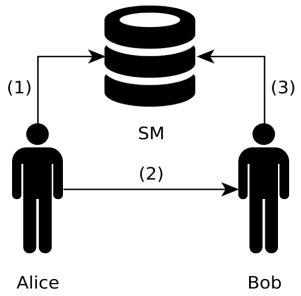

Figure 3. IBE emulation architecture

Formal definitions of our approach are given in Figure 4. As a first step, Alice has to retrieve the recipient's (*Bob's*) public key rk_{pub} by providing a set of attributes $attrs$ (identifying *Bob*) to equation (5) (step (1) in Figure 3). Here, the SM decides if it needs to create a fresh key pair or if a suitable key pair is already available. *Alice*, however, does not see any difference in the workflow. Note that, in comparison to the first step of the original IBE/ABE scheme i. e. equation (1), the call only receives some additional parameters. The result is also a public key. After the initial step, *Alice* has all the data she needs to encrypt some plain payload data p by the means of equation (6) and to send the resulting cipher text c to *Bob* (step (2) in Figure 3). Compared to the IBE/ABE workflow, there is no need to include the attributes $attrs$ anymore,

$$getKey(attrs) = rk_{pub} \quad (5)$$
$$encrypt(rk_{pub}, p) = c \quad (6)$$
$$derive(attrs) = rk_{priv} \quad (7)$$
$$decrypt(rk_{priv}, c) = p \quad (8)$$

Figure 4. IBE/ABE emulation definitions

since they are inherent with the key. *Bob* receives the cipher text c and requests the central SM to decrypt the data (step (3) in Figure 3). Before performing the request, the central SM verifies if *Bob* has all required attributes and gains the recipient's private key rk_{priv} through equation (7). Note that this step is reduced to merely find the private key rk_{priv} of *Bob* compared to the original IBE/ABE workflow. Furthermore, the result is not returned back to *Bob* but is instead used to perform the next step. In case *Bob* is indeed authorized to use his private key rk_{priv}, the SM executes equation (8) and sends the plain payload data p back to *Bob*. Equation (8) does differ from the original equation (4) by the means of not requiring the public master key.

In order to omit sending the encrypted data to the central SM, one can deploy hybrid encryption schemes, as used with CMS. In that case, the cipher text c from above is merely the content encryption key (CEK). The content encryption key preferably is a comparably short symmetric key (AES-256 for example) and can be used to bulk-encrypt large amounts of data. As the CEK is a part of the data block sent to the recipient, no additional tasks have to be performed except for wrapping (encrypting) the key with the recipients public key rk_{pub} before sending and unwrapping the CEK using the recipient's private key rk_{priv} to gain the plain payload data p. While this extension offers a performance gain since the payload data does not have to be sent to the central SM, the SM does not see the plain payload data p directly and thus, has a harder time eavesdropping on the actual data.

All in all, our approach of bringing IBE/ABE concepts to today's infrastructures does not require new players in the application scenario. However, it exchanges the central entity PKG with our SM. Furthermore, the communication lines remain pretty much as they are in the IBE/ABE scheme, only the transmitted data is slightly different. I.e. the first step returns a different public key. The hybrid encryption scheme extension is applicable to the original IBE/ABE scheme as well.

V. Implementation

In order to evaluate our proposal, we developed a prototypical implementation based on the CrySIL architecture [16]. CrySIL offers a (semi-) centralised key storage that can be deployed to be publicly accessible and has built-in authentication features available. We complemented CrySIL's easy-to-extend architecture by a key store implementation that features on-demand key creation and directory service functionality besides lifetime key management, protection, and key usage policies. The ability to deploy the CrySIL key storage on a smart phone [15] enables us to get rid of the trusted third party SM.

In this section, we give a brief summary of CrySIL, discuss the custom key store as well as give a detailed work flow description.

A. Cryptographic Service Interoperability Layer

The Crypto Service Interoperability Layer (CrySIL) [16] concept has been created to meet today's heterogeneous device landscape and allows the user to use her keys within cryptographic operations, regardless where the key resides and where it is needed. In a nutshell, the user takes one of her devices and launches an application to perform some cryptographic task. The application interfaces with the interoperability layer, CrySIL, which connects to another device. This other device has access to the actual cryptographic primitive, creates and validate authentication challenges if required and performs the requested operation. The result is returned to the interoperability layer and back to the application running on the device of the user. A graphical illustration of the workflow is given in Figure 5.

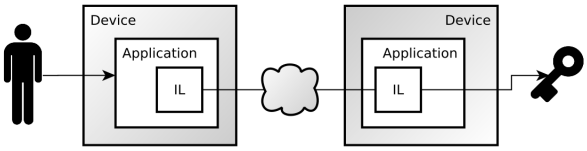

Figure 5. CrySIL's basic architecture

Most conventional cryptographic service providers receive commands, perform the required actions, and return the result to the caller. To achieve CrySIL's interoperability goal, CrySIL breaks the classic cryptographic provider apart. The resulting parts – modules – have different jobs and work together to form the actual cryptographic service provider. *Receivers, actors*, a *router*, and other modules handle inter-node communication, protocol mappings, crypto, advanced crypto and authentication. A *receiver* for example acts as a protocol bridge to the interoperability layer with pre-build implementations for Java's Cryptographic Extension (JCE), the Windows CNG, or the W3C Web Cryptography API as well as PKCS#11 and a SOAP-based web interface. An *actor* makes the services of a crypto provider, e.g. a smart card or an HSM, available to the interoperability layer and collects authentication information as they are required. An actor+ performs advanced cryptographic tasks e. g. CMS. The *router* connects the modules together. Anyhow, all modules are considered as building blocks and are not restricted to any technology, platform, or programming language. Every configuration of a router and other modules is referred to as a *CrySIL node* and forms the heart of the concept. The modular node design enables the flexibility and extensibility of the concept while keeping the overall architecture simple. An illustration of modules and their interconnections is given in Figure 6.

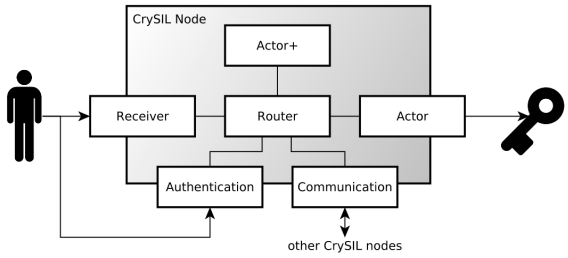

Figure 6. CrySIL node architecture overview

Cloud-scenarios with multi-device users require for cryptographic primitives and services to be available anywhere and at any time. CrySIL's answer is transparent off-device cryptography, which mandates inter-node communication. The *communication* modules are in charge of inter-node communication. They simply take a request and send it to another off-device communications module. The most basic implementation is HTTP(s). Yet, arbitrary transport protocols, such as HTML5 Web Messaging, Web-sockets, or IPSec are suitable.

The off-device crypto feature of CrySIL requires for key usage constraints as the keys are potentially available to the public. As CrySIL can interface with key providers that may already require for authentication information, CrySIL has to collect and provide the authentication data. Within CrySIL only the *actor* knows about the authentication requirements of its key provider and has to challenge the user accordingly. CrySIL can support a multitude of authentication methods, such as simple PIN challenges, external identity providers (OAuth, OpenID Connect) or multi-factor authentication methods. The CrySIL infrastructure gathers authentication data from the user as well.

The drawback of having the need for yet another trusted third party is addressed by the MoCrySIL extension [15]. MoCrySIL allows for operating a CrySIL node on a users mobile phone. Modern smart phones do have hardware security modules available and therefore offer pretty good key protection. Furthermore, having the key service at the user's fingertips lets the user directly confirm or deny incoming crypto requests and thus, strengthen the security and controllability of the infrastructure.

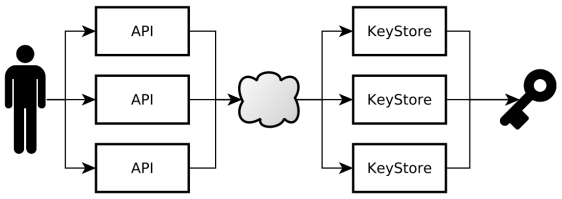

Figure 7. Interoperability architecture view

Putting the pieces together, CrySIL renders off-device crypto completely transparent to the user, the developer, and to the application while maintaining a simple architecture. The resulting architecture is depicted in Figure 7. A user benefits from her ability to use a variety of keys within different cryptographic operations provided by different crypto providers from a variety of applications on different devices.

B. Actor/Key Service

CrySIL already includes a wide set of the needed features but lacks in providing a key service/actor that offers dynamic key creation and a directory service. Therefore, we complemented the interoperability layer with a new actor/key service (in the following referred to as *actor*).

The *actor* is composed of four distinct building blocks. The first block is the key database. The key database is the place where the actor looks for suitable keys and where it stores keys that are newly created. In our prototypical implementation, this key database is implemented as a software security module. CrySIL, however, enables the use of plain HSM solutions as well as hybrid solutions where keys are wrapped by an HSM and stored to disk (encrypted). The key database maps usage constraints, i. e. policies, to the respective keys.

The second major building block is an Identity Provider (IdP) or Attribute Provider (AP). These providers are used to provide the proof that a certain user possesses certain attributes. The attributes are then matched against the key policies to authorize requests. For our prototypical implementation we integrated Microsoft's Active Directory (AD) via the Lightweight Directory Access Protocol (LDAP), serving as IdP for assessing a users identity and as an AP to provide further attributes, such as a *network domain* or *workgroup*. By chosing AD, we tailored the *actor* to business use cases. For deployment scenarios that do not have AD available, other IdPs/APs such as Facebook[4] or the STORK framework[5] can be used as well.

The third building block is a certification authority (CA). The CA is capable of creating new X.509 certificates for encryption usage. The actor simply issues a certificate signing request (CSR) towards the CA and awaits its answer. This clean interface allows for a variety of CA types to be used. For our prototypical use case we integrated a company-owned CA. Other deployment scenarios can use public CAs as well as home grown solutions like self-signed certificates.

The fourth and last building block of the *actor* is, of course, a business logic. This logic links the building blocks with each other, interprets crypto requests, enforces policies, and acts as a directory service frontend.

The combination of all building blocks allows for a directory service feature as well as for the dynamic key creation feature. For our implementation, we limited our attribute selection capabilities to the recipient's email address. As we solely focussed on IBE emulation, the identity-based authentication method of CrySIL suffices for this task. For other deployment scenarios, it is conceivable to enhance CrySIL in order to support advanced attribute selection, for example, with predefined attribute lists where the sender can select the subset that fits her needs. The key policies stored in the database are filtered for the given attributes and the resulting list is returned to the caller.

The dynamic key creation feature kicks in when there is no matching key for the given attributes (the email address in our case). A new key pair/certificate is created and sent to the CA for certification. Then, the key is tagged with policies that match the given attributes. The whole structure is stored in the key database for future use. In case, there is a directory service request ongoing, the query is repeated and now yields the freshly created key/certificate.

C. Workflow

For a detailed workflow description we assume a simple IBE/ABE scenario with a sending entity *Alice*, a receiving *group* with members *John*, *Su*e, and *Bob*, and the central *SM* having the business logic, IdP/APs, the CA, and the key database ready. An illustration of the setup is given in Figure 8. Communication with the SM is done by the interoperability layer CrySIL. The goal is for *Alice* to send some encrypted data to the *group* without the need to preshare a key amongst the group.

[4]https://www.facebook.com/
[5]https://www.eid-stork2.eu

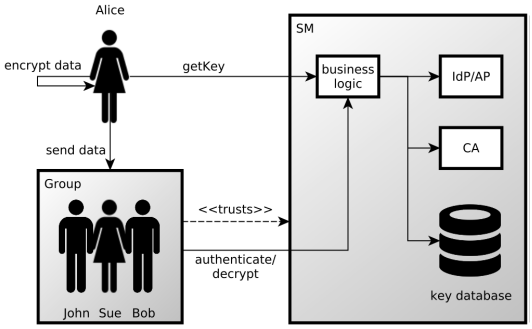

Figure 8. Deployment scenario

First, *Alice* states her intention about wanting to encrypt some data for the attributes $attrs = \{group\}$ to the central SM following equation (5) (*getKey* in Figure 8). Now, the business logic checks its key database whether it finds a suitable recipient key $rk\,with\,attrs = attrs(rk)$. Upon success, the public key rk_{pub} is returned to answer the *getKey* request. In case there is no suitable recipient key available, a new key pair/certificate is created, certified by the CA, gets a policy attached that matches the recipient ($attrs \in \{group\}$), and is stored in the key database and returned to answer the *getKey* request.

Subsequently, *Alice* uses the received public key rk_{pub} to perform a hybrid encryption of the plain payload data, i.e. create a content encryption key (CEK), use it to encrypt the plain payload data, and wrap the CEK with the recipient key rk_{pub} by executing equation (6). Alice can now send the data structure containing the encrypted payload data, the wrapped CEK and required attributes to the recipient.

Any member of the group *group* can now initiate a decryption process. *Bob*, for instance, contacts the SM and expresses his intention to unwrap the CEK he received from *Alice*. The business logic of the *SM* searches for the wrapping key (indicated by the given attributes) following equation (7). Note that the key in question now exists since, if it had not existed at the time *Alice* initiated the encryption process, it would have been created during the encryption process. Then, however, the *SM* initiates one or more authentication processes in order to collect proof about *Bob* having certain attributes. The business logic queries the IdPs/APs with the authentication information he was provided with by *Bob*, if anything was correct, gets proof that *Bob* indeed has the required attributes.

After successful retrieval of the recipient key and successful authorization of the user, the central SM performs the decryption process by following equation (8) and returning the plain CEK to *Bob*. *Bob* can now reverse the hybrid encryption scheme by decrypting the payload cipher text using the CEK he just received from the SM.

VI. Evaluation

In this section we give a functional analysis and compare our solution to the original IBE/ABE concept. We also give a thorough security analysis and risk assessment for different deployment scenarios with countermeasures as well.

A. Functional

CrySIL – with our extensions – offers similar functionality as IBE/ABE concepts do. The core feature of our approach is the on-demand key creation. Policies are assigned to the fresh keys. These policies constrain key usage to certain methods (encrypt/decrypt), users, or in general to attributes. Only when a user can prove that he indeed owns the requested set of attributes he is allowed to perform a certain operation. This does also allow encrypting for recipients that do not even exist yet.

Having IBE/ABE concepts available for today's popular PKI infrastructures is, however, the main contribution of this work. The CrySIL infrastructure is designed for X.509-based cryptographic primitive representations. Thus, our approach fits into today's X.509-based IT, such as email or data encryption solutions. However, one culprit remains: in general, there is no consistent on-demand key lookup procedure built into available client applications. The most promising approach would be to provide the directory service feature via LDAP since some clients are designed to retrieve information via LDAP.

Although our protoype is targeted to emulate IBE only, ABE comes almost for free once the system learns to support advanced key policies. Classic attributes like email address or workgroups can be accompanied by attributes that handle date/time constraints, geographical constraints or similar. Of course, with a great number of attributes another challenge is introduced. For now, there has to be one key for a distinct set of attributes. That might result in a whole lot of key pairs. In practice, this might not be as much of a drawback for reasonable sized groups of attributes because the SM can be operated in the cloud and the cloud is capable of managing huge numbers of keys. With huge numbers of keys, yet another challenge has to be faced, namely when to get rid of them. The system is not designed for being maintained manually, so there has to be a sort of cleanup mechanisms. Intuitively, a "use once and remove" policy might be suitable for IBE and single recipients. However, it might not be adequate for scenarios with group-based ABE. The approach we use in our prototype creates keys (i.e. certificates) that are valid for a certain time. When the time passes, the keys can be removed. This, however, is target of future work.

With CrySIL, even hierarchical IBE (HIBE) [10] can be emulated. Hierarchical IBE was designed to meet delegation needs, for example, when the PC stays in the office while the employee is on a business trip – i.e. when a slave identity (a laptop) needs to act on behalf of the master identity (a PC). CrySIL does something quite similar since it is designed to meet the demands of today's multi-device users. Different devices such as smart phones, tablets, laptops, or browsers have different key protection capabilities and, hence, it is not reasonable to copy one's private key to every single device. When compared to HIBE, seeing the central SM as master identity, and any devices of the user as slave identities, the behavior is quite the same.

Finally, our approach is suitable for different deployment scenarios. First, the system can be deployed to meet enter-

prise use cases **(S1)**. Enterprises may have AD intrastructures available to fit the roles of the IdP and AP, performing user authentication and attribute provision. Enterprises may also have in-house CA solutions available for on-demand key creation. In total, communication security policies would benefit greatly from the automated key/certificate creation offered by our approach.

Another possible deployment scenario is an (emulated) IBE/ABE service for people around the world **(S2)**. For authentication and attribute source the service can leverage established services, such as Facebook or eID (STORK). Keys created on demand can be certified by public CAs.

And last but not least, one can deploy our approach as a personal service either at home or on a smart phone **(S3)**. Facebook and eID, or even simple username/password authentication can be used for the roles of IdP and AP. Self-signed keys that were created on demand may also suffice for many use cases.

B. Security

The goal of the security assessment is to understand the assets, threats, possible countermeasures and residual risks which lets us compare the results with the IBE/ABE concept. A tabular overview of threats, assets, countermeasures and applicable scenarios is provided in Table I.

For our security assessment, we make the following assumptions. First, the CA is trustworthy and behaves exactly as it should. In case the CA is attacked or rogue, the consequences are the same as in the classic PKI scenario and are therefore considered out of scope. Second, communication lines are considered secure, i.e. there is no eavesdropping or tampering possible. We rely on the security of the CrySIL infrastructure, already assessed in the respective literature [16].

We identified three main assets in our setup. First, the payload data **(A1)**, e.g. the data that to be transmitted securely between sender and recipient, is considered as an asset. However, disclosure of the data would only compromise the data itself and, thus, has only little effect on the overall system. Second, disclosure of the private key data **(A2)** of the recipient would render a whole communication with this recipient compromised. Note that a recipient can also be a group of entities that share a common attribute. If the respective private key is disclosed, the communication of the whole group is compromised. Last but not least, a recipient's authentication data **(A3)** is used to prove that a recipient owns certain attributes and consequently, is allowed to employ a respective key. Compromised authentication data would therefore render any communication compromised that is encrypted towards this special attribute even if the underlying key is renewed.

The obvious main threat is a rogue central SM as it manages the recipients private keys. First, the SM can use the recipient's private keys to eavesdrop on data **(T1)**. For example, the SM can authorize itself to perform cryptographic operations. The assets under attack are thereby A1 and A2. Second, the SM might disclose a recipient's private key **(T2)**. A disclosed private key would compromise A2 and consequently A1 and therefore has major impact on the security of the system. Last but not least, the central SM could issue a key tailored to a different attribute set than requested in the initial fetch key operation **(T3)**. In practice that would change the recipient, without the sender knowing. This threat again would compromise A2 and therefore A1.

A rogue IdP or AP can compromise security as well. First, a rogue IdP or AP can grant an attacker attributes that he actually does not own **(T4)**. Such an attack has a similar effect as compromising A3 and therefore A1. Second, a rogue IdP or AP can disclose authentication data **(T5)** and therefore compromise A3 and thus A1 as well.

A sender generally does not have an interest on attacking the system, as he only wishes to send data securely to a recipient. However, a rogue sender can team up with an involved IdP/AP and compromise the system by adopting the role of the attacker in T4. As in T4, the assets A3 and A1 are compromised.

A recipient generally does not have an interest on attacking the system neither, as he only wants to receive data securely from a sender. The receiver, though not having an obvious advantage, can disclose his own, authentication data and therefore compromise A3.

threat	asset under attack	applicable countermeasures	applicable scenario
T1	A1, A2	C2, C3	S2
T2	A1, A2	C1, C2, C3	S2
T3	A1, A2	C2, C3	S2
T4	A1, A3	C2, C3, C4	S1, S2, S3
T5	A1, A3	C2, C3, C4	S1, S2, S3

Table I
RISK ASSESSMENT OVERVIEW

As a remedy for some threats, countermeasures are available. The use of a hardware security module **(C1)** where keys are protected throughout their lifetime, tackles the disclosure of private key data T2. Deployment of MoCrySIL, i.e. having the private keys on one's smart phone **(C2)** is another countermeasure. This action, of course, depends on the (active) smart phone security features (e.g. whether hardware-backed key storage capabilities are available). Assuming a perfectly secure smart phone, the remedy tackles T1, T2 and T3 as the SM becomes trusted. T4 and T5 can be managed by handling an individual IdP/AP implementations **(C4)** (e.g. username/password authentication that does not rely on another third party). Last but not least, using hybrid encryption schemes **(C3)** makes disclosing and eavesdropping the payload data A1 harder and is, hence, recommended.

Last but not least, for the risk assessment, the deployment scenarios (defined toward the end of section VI-A) must be taken into account. The first scenario S1, the enterprise scenario, has its own and therefore trusted central SM. Therefore, the threats T1, T2, and T3 do not apply to this scenario. Neither do these threats apply to the scenario S3, where the user operates her own and therefore trusted central SM. The cloud service scenario S2, however, is at risk for the threats T1, T2, and T3. Note that an enterprise that outsources the SM operation is to be counted as S2 as well. The IdP/AP related threats T4 and T5 apply to any scenario when external IdPs/APs are used. Hence, operating a private IdP/AP (e. g.

AD), i. e. C4, reduces the residual risk significantly. Generally, trust in a state-owned eID as IdP/AP might be more reasonable than trusting data coming from Facebook for example.

VII. Conclusions

Overall, we succeeded in emulating the IBE/ABE concepts for PKI/X.509-based IT infrastructures without the lack of features and loss of security.

With CrySIL and its centralized key service deployment scenario we made keys accessible from everywhere and at any time, without affecting security due to weak key protection capabilities of clients (e.g. browsers). That enables for HIBE emulation as well.

For security and privacy, we found that rogue SMs and IdPs/APs are capable of compromising the whole system. That is, however, quite similar to a rogue PKG service within the original IBE/ABE scenarios. The threat of a rogue SM returning a wrong public key to the sender (T3), however, does not apply to the original system, because the key is computed by the client.

We stated three deployment scenarios that target large and medium sized enterprises, classic cloud service deployment, and personal deployment. Our prototype is tailored to fit the enterprise scenario and eases administrative tasks for fulfilling a enterprise-wide encrypted communication policy well. The cloud deployment scenario, however, does not give great advantages in security and trust compared to classic PKI solutions, it does, however, enable encrypted communication with recipients, that do not have certificates available. Some crypto does push privacy and security more then no crypto at all.

As for future work, some challenges remain. First, as client applications tend to not fetch certificates on demand, our approach is somewhat handicapped. Future work, therefore, is to implement an LDAP interface to the SM for the initial getKey command, as there are some key application that do live lookup of certificates. And of course, a more robust and consistent solution for on-demand certificate lookup must be found. Another open issue, in need for attention, is the number of keys that appear when using big attribute spaces.

References

[1] Man Ho Au, Qiong Huang, Joseph K. Liu, Willy Susilo, Duncan S. Wong, and Guomin Yang. Traceable and retrievable identity-based encryption. In *Proceedings of the 6th International Conference on Applied Cryptography and Network Security*, ACNS'08, pages 94–110, Berlin, Heidelberg, 2008. Springer-Verlag.

[2] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer Berlin Heidelberg, 2001.

[3] Melissa Chase. Multi-authority Attribute Based Encryption. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, volume 4392 of *Lecture Notes in Computer Science*, pages 515–534. Springer, 2007.

[4] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *Cryptography and Coding*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer Berlin Heidelberg, 2001.

[5] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, Internet Engineering Task Force (IETF), May 2008.

[6] Carl Ellison and Bruce Schneier. Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure. *Computer Security Journal*, 16(1):1–7, 2000.

[7] Vipul Goyal. Reducing trust in the pkg in identity based cryptosystems. In *Advances in Cryptology - CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 430–447. Springer Berlin Heidelberg, 2007.

[8] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based Encryption for Fine-grained Access Control of Encrypted Data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 89–98. ACM, 2006.

[9] Peter Gutmann. PKI: It's not dead, just resting. *Computer*, 35(8):41–49, 2002.

[10] Jeremy Horwitz and Ben Lynn. Toward Hierarchical Identity-Based Encryption. In LarsR. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481. Springer Berlin Heidelberg, 2002.

[11] H. Leitold, A. Hollosi, and R. Posch. Security architecture of the Austrian citizen card concept. *18th Annual Computer Security Applications Conference, 2002. Proceedings.*, 2002.

[12] Allison Lewko and Brent Waters. Decentralizing attribute-based encryption. In KennethG. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 568–588. Springer Berlin Heidelberg, 2011.

[13] Sascha Müller, Stefan Katzenbeisser, and Claudia Eckert. Distributed attribute-based encryption. In *in International Conference on Information Security and Cryptology (ICISC08)*, pages 20–36. Springer, 2008.

[14] Clemens Orthacker, Martin Centner, and Christian Kittl. Qualified Mobile Server Signature. In *Proceedings of the 25th TC 11 International Information Security Conference SEC 2010*, 2010.

[15] Florian Reimair, Peter Teufl, Johannes Feichtner, and Christian Kollmann. MoCrySIL - Carry your Cryptographic keys in your pocket. In *Proceedings of seCrypt*, 2015.

[16] Florian Reimair, Peter Teufl, and Thomas Zefferer. WebCrySIL - Web Cryptographic Service Interoperability Layer. In *Web Information Systems and Technologies*, 2015.

[17] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology–EUROCRYPT 2005*, pages 457–473. Springer Berlin Heidelberg, 2005.

[18] Ryuichi Sakai and Masao Kasahara. Id based cryptosystems with pairing on elliptic curve. *IACR Cryptology ePrint Archive*, 2003:54, 2003.

[19] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer Berlin Heidelberg, 1985.

[20] Steve Sheng, Levi Broderick, Jeremy J Hyland, and Colleen Alison Koranda. Why Johnny still can't encrypt: evaluating the usability of email encryption software. In *Symposium On Usable Privacy and Security*, pages 3–4, 2006.

[21] Sriramkrishnan Srinivasan. Identity based encryption: Progress and challenges. *Information Security Technical Report*, 15(1):33 – 40, 2010. Protocols and Cryptography.

[22] Nevena Vratonjic, Julien Freudiger, Vincent Bindschaedler, Nevena Vratonjic, Julien Freudiger, Vincent Bindschaedler, Jean-pierre Hubaux, Nevena Vratonjic, Julien Freudiger, Vincent Bindschaedler, and Jean-pierre Hubaux. The Inconvenient Truth about Web Certificates. In *Proceedings of the 10th Workshop on Economics of Information Security*, pages 11–14, 2011.

[23] Alma Whitten and J.D. Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. *Proceedings of the 8th USENIX Security Symposium*, pages 169–184, 1999.

[24] P. Yee. Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 6818, Internet Engineering Task Force (IETF), January 2013.