RAD HAZU. MATEMATIČKE ZNANOSTI
Vol. 25 = 546 (2021): 143-159
DOI: https://doi.org/10.21857/y54jofkjdm

FORMAL LANGUAGE IDENTITY-BASED CRYPTOGRAPHY

Ádám Vécsi and Attila Pethő

Abstract. The rapid growth of the digital economy makes fine-grained access control more and more challenging. One of the most impacted areas is cloud computing, which for security purposes requires cryptographic access control. Currently, the best solution for that is the use of Attribute-based Cryptography, which allows the definition of access policies, based on the attributes of entities. Unfortunately, this family of schemes comes along with a significant drawback, specifically the required user-side computation is growing with the growth of the complexity of the access policy. We provide a concept, called Formal Language Identity-based Cryptography, which gives a solution to this problem, making fine-grained cryptographic access control practical.

1. Introduction

Identity-based Cryptography (IBC) is a unique branch of public-key cryptography. This uniqueness lies in the fact that the public key in these systems is a known identifier string of an entity. One may think about an email address, a username, or a phone number. The concept behind IBC was coined by Adi Shamir [20]. The core idea of IBC was to simplify the certificate management and eliminate the need for certification authorities, that is why the known identifier strings as public keys came handy. In the public key infrastructure scenario, public keys and user identities are bound together with certificates. With IBC, however, there is no need for such certificates, since the public key corresponds directly to the user identity. However, at this point, Shamir could not build an encryption scheme. The first break-through was the Boneh-Franklin Identity-based Encryption system [3], having the novelty of using the operation of pairing to achieve the needed performance for practical use. This step forward opened the gate for this branch of cryptography.

Furthermore, there is an expansion of the idea that the public key may contain more information and domain-specific data, not just the identity of the

2020 *Mathematics Subject Classification.* 94A62, 68P25.
Key words and phrases. Identity-based Cryptography, Attribute-based Cryptography, access control, proxy signature.

user, for example, the current date. This extension enables a broad-spectrum of use-cases [21,23]. Although, one limitation of IBC is that the public key of the receiver must be bit-accurate to the encryption key to be able to extract the belonging private key. Consequently, IBC is not able to handle finely granulated policies as keys.

In multi-user computer systems, it was always necessary to define different levels of access, so the users are only authorized to the proper resources. It is handled by access control services, which mostly are policy-based. The owner of the resources defines and manages authorization policies that specify which user or which role has access. One of the most dominant multi-user systems is cloud computing, which is applied in many commercial fields with numerous users. The fast growth of cloud service providers makes it mandatory to maintain scalable, reliable, and flexible access control.

There are several attempts to implement flexible and fine-grained access control mechanisms [4, 11], but security is still one of the main issues. There are two aspects of security issues. One is the data breach caused by hacking or identity management issues [16], what is preventable with secure authentication protocols [9, 10]. The other is that it is necessary to assume that the providers are honest but curious, which means they may try to peek into the user data to extract information without the knowledge of the user. Because of this issue, cloud service providers can not be regarded as trustworthy entities. A solution to this problem seems to be the usage of cryptographic methods to outsource encrypted data only.

The most convenient option is Attribute-based Encryption [2,7], to handle the encryption and the access control simultaneously. This type of cryptosystem uses an access policy to determine which ciphertexts a user able to decrypt. The core idea is to treat the keys as expressions, which contain logical operators between attributes and values, thereby the keys are more flexible and capable to define fine-grained access policies. Although, a significant drawback of the ABC schemes is that they require more computation on the user-side (encryption and decryption functions) with the growth of the complexity of the access policy. This in turn affects the usability of these schemes, since several potential applications target devices with limited computational power.

With encryption, the user can protect its data, but there is still the issue of placing unauthentic data to the cloud servers. Most of the solutions to this problem require a lot of additional client-side computations to provide authentication, which in most cases, especially for mobile data owners is impractical. An obvious solution to this is the proxy signature [13], which allows the delegation of the signing operations to computationally powerful entities, providing authenticity to the data with signature.

Our motivation was to design a cryptographic system with the possibility of fine-grained access policy definition, but without the performance drawback

on the client-side, to provide a cryptographic access control method, which is usable in practice.

The model we designed is based on IBC, thus we inherit a system where every entity has a public key, which is an identity and some linked domain-specific data. In the standard IBE model, this public key is also the encryption key. In our scheme, however, that is not the case. The main ideas of our model are the introduction of an authorization formula, which is characterized by the encryptor entity as an access policy, and that the public keys are used as authorization keys. Hence everyone whose public key satisfies the authorization formula is authorized to extract the belonging decryption key. This inspection is handled by the trusted Private Key Generator, which is also inherited from IBE. The keys used for encryption and decryption are still valid IBE key pairs.

Furthermore, to utilize the potential of the authorization formula, we use formal languages. The formula is built from attribute constraints, concatenated with logical operators. Each attribute constraint comprises a formal language, defining which entities may have access and which may not. Hence, the scheme provides robust tools for defining a fine-grained access policy targeting an arbitrary group of entities.

Using this idea, we built encryption and multi-proxy signature schemes to provide a solution to both of the mentioned cloud computing issues, namely the data leakage and data authenticity. Our concepts are built on secure Identity-based Encryption (IBE) and Identity-based Signature (IBS). Besides those, the only operations required are some formal language operations and the evaluation of logical formulas. Thus the level of security of our schemes is the same as the used IBE and IBS.

The rest of the paper is organized as follows. Section 2 contains some general definitions about Identity-based Cryptography, Proxy Signature and formal languages. The related work is found in Section 3. Afterward, Section 4 presents some necessary definitions and our schemes. Section 5 outlines the performance of our concept, compared to the BSW-ABE [2]. Finally, Section 6 gives a conclusion.

2. Preliminaries

2.1. *Identity-based Cryptography.* In the Identity-based Cryptography model, there are two main roles distinguished by their purpose. There are the user entities (encryptors, decryptors, signers and verifiers) and a trusted third party called Private Key Generator (PKG). The PKG generates the private key for every request, based on the user ID, and some system parameters, and also on the master secret, which is only known by the PKG.

An IBE scheme consists of four algorithms:

- **Setup**: Based on the given security level parameter λ, this function outputs the public parameters of the system $params$ and a master secret ms.
- **Extract**: Receives a public key (user ID) id and the master secret ms and outputs a private key pk_{id}.
- **Encrypt**: Given the public parameters of the system $params$, a public key id and a message m, the function outputs a ciphertext c.
- **Decrypt**: Receives a private key pk_{id} and a ciphertext c and outputs the plain message m.

The Encrypt and Decrypt functions form the inverse of each other, which means if there is a message space M, then

$$\forall m \in M : Decrypt(Encrypt(m, id), pk_{id}) = m.$$

The IBS model is also similar, just instead of the Encrypt and Decrypt functions, it consists a Sign and a Verify function.

2.2. *Proxy signature.* The idea of proxy signature schemes is to delegate the task of digital signature creation. The main reasons for this solution are that the original signer may be offline regularly, which means he/she cannot perform this task, or the original signer does not have enough computational power. This way the task can be delegated to a proxy signer, who helps out the original signer with his computational power and availability.

There are several types of proxy signatures based on the delegation of the signing key. The main types are full delegation, partial delegation, and delegation by warrant. I would also mention another useful type, namely partial delegation with warrant [12].

Let us say Alice is the original signer and Bob is the proxy signer. In the case of full delegation, Alice's secret key is given to Bob. So Bob has the same capabilities as Alice. In practice, this method is impractical and insecure, because the signatures created by Bob and Alice, are indistinguishable and Bob is not limited in any way.

In partial delegation, Bob receives a signing key which differs from Alice's. Hence the signatures created by Bob are different from Alice's. With this, the signatures are distinguishable, but there is still no restrictions on Bob's signing capability.

The weakness of no limitation is eliminated by the delegation by warrant technic, because in the warrant Alice specifies, what kinds of messages are allowed to sign by Bob. The warrant may contain any kind of description about the limitations of Bob and it is signed by Alice. The warrant also contains its valid period.

While the key revocation is not a problem in the case of delegation by warrant, because of the included validity period, it is for the partial delegation, which requires a revocation method, which makes those schemes more

difficult and complicated. Here comes to the picture, the partial delegation with warrant, which is the combination of the last two technics.

2.3. *Formal Language.* Formal language is a powerful and flexible tool to define the required set of strings. Here we collect the basic definitions, which help to the understanding of the paper. You find a good overview of the classical theory of formal languages in Salomaa's work [19].

DEFINITION. ***Alphabet.*** *A finite but nonempty set of symbols (or letters). Alphabets are usually denoted as* Σ.

DEFINITION. ***Word.*** *A finite sequence of symbols chosen from a given alphabet* Σ. *The set of all words that can be formed using symbols from* Σ *is denoted as* Σ^*.

DEFINITION. ***Word over alphabet.*** *If we form a word using symbols from a given alphabet* Σ, *then this word is said to be a word over* Σ.

DEFINITION. ***Language.*** *Given an alphabet* Σ, *a language* L *is a set of words over* Σ: $L \subseteq \Sigma^*$.

DEFINITION. ***Grammars.*** *A finite set of rewrite rules generating words over some alphabet. Grammars are usually denoted as* G *in which the language generated by the grammar is written as* $L(G)$.

3. Related Work

There are multiple directions and attempts to achieve a broadcast type of encryption in the Identity-based and Attribute-based domain. Some of them even focus on the fine-grained definition of the target group.

The first step towards was the Fuzzy-IBE [17]. The main novelty of this scheme found in the public keys, which are not just strings, but sets of attributes. With this concept, it could add fuzziness into the system in a way that it requires only a certain amount of overlap between the encryptor key and the public key of the decryptor, not a total bit accuracy. The advantage of this system is much faster group encryption, but it comes with runtime drawbacks in every other function (setup, extract, and decrypt).

Then came the general idea with the definition of a big target set of entities, who are aimed at the broadcast encryption [5, 18, 25]. The research of this type of scheme is focusing on constant-size ciphertext and private keys regardless of the size of the target group [5, 25], which is an important area, but the performance of the encryption and/or the decryption functions is highly dependent on the size of the target group. Also, the maintenance of the target group might be a time-costly process, if the group is big.

Another direction towards the fine-grained definition of the encryption targets is Identity-based encryption with wildcards (WIBE) [1]. The idea of this attempt is that the encryption key is a pattern, which is treated as a

vector of bitstrings and wildcards. Let P be the encryption key, then $P = (P_1, ..., P_l) \in (\{0,1\}^* \cup \{*\})^l$. Here the wildcard character $(*)$ represents an "any kind of bitstring" part in the key.

Example 3.1. **@cs.*.edu*

The example 3.1 pattern represents that anyone who has a computer science educational email address in that form can decrypt the encrypted document. A valid email address is *name@cs.unideb.edu*. This scheme allows us to use a very flexible encryptor key, although if too many wildcards are added, it slows down excessively.

Traditionally identification and authorization are different tasks, the first is cryptographic, while the second is administrative. The concept of the Fuzzy-IBE that the public keys are sets of attributes started a new branch called attribute-based cryptography, which serves, among others, with a unique solution for identification and authorization. This branch follows the same idea, but with an extension of it, with building an access-tree to the keys. In the leaf nodes, placed the attributes and the other nodes are logical gates of AND, OR operations. There are two main types of attribute-based cryptography, key-policy [7], and ciphertext-policy [2]. The difference is located in which data is associated with policies and which with sets of descriptive attributes. The key-policy is working with ciphertexts associated with sets of attributes and the user public key with policies, while ciphertext-policy is the opposite. There are also more advanced attribute-based cryptography schemes, which allow more flexible access-trees, with the occurrence of negation [14,15,22] and multi-use of attributes [22]. However, a significant drawback of the Attribute-based Cryptography schemes is that they require more computation on the user-side functions (encryption and decryption), with the growth of the complexity of the access policy. This directly affects the usability of these schemes since several potential applications target devices with limited computational power. The authors in the work [22] were able to create a scheme that managed to fix this issue, but only for decryption, and only if the access-tree is built with AND and OR gates and does not contain multi-use, nor negation of the same attribute. It is a step further, but it involves less access-policy flexibility.

4. Formal Language Identity-based Cryptography

Now, we introduce our concept to fine-grained cryptographic access control. These schemes provide an especially flexible access policy definition, with the usage of formal languages and without the performance drawback of the client-side algorithms, making them effective in practice.

4.1. *Definitions.* First, we define the notions needed for our schemes.

DEFINITION. ***Attribute***: *Let $\sum$ be an alphabet specified by the domain our schemes are applied in (most commonly the alphabet of a natural language). An attribute a is a pair (N_a, L_a), where N_a is a word over $\sum$ for the name of the attribute and L_a is the formal language containing all the possible values of the attribute. The universe of the attributes U is also limited only by the domain.*

DEFINITION. ***Property***: *A property $p = (N_p, V_p)$ defines actual values of an entity regarding to the attribute a, if N_p is a word over $\sum$ such that $N_p = N_a$ and V_p is a formal language such that $V_p \subseteq L_a$.*

DEFINITION. ***Attribute constraint***: *Let $a = (N_a, L_a)$ be an attribute. $\Gamma = (N_\Gamma, L_\Gamma)$ is an attribute constraint for a, if $N_\Gamma = N_a$ and $L_\Gamma \subseteq L_a$, where L_Γ is the formal language, which contains the accepted property values.*

DEFINITION. ***Authorization formula***: *An authorization formula Υ is a logic formula that contains attribute constraints and boolean operators.*

DEFINITION. ***Public key***: *Let Ω be the set of entities and P the set of properties in a given domain. In our scheme PK_α is the public key of an entity $\alpha \in \Omega$, if $PK_\alpha \subseteq P$ and $\forall \beta \in \Omega$, $\exists\ ID_\alpha \subseteq PK_\alpha$, such that $ID_\alpha \setminus ID_\beta \neq \emptyset$. ID_α is called the **identifier** of α.*

The schemes are based on secure Identity-based Encryption (IBE) and Identity-based Signature (IBS). To be clear with the notations, and the purpose of the algorithms, here we will redefine the IBE algorithms already mentioned in section 2 and define the IBS algorithms, which will be referred to in our schemes.

An IBE scheme is usually specified by four algorithms: {*SetupIBE*, *ExtractIBE*, *EncryptIBE*, *DecryptIBE*}, where

- **SetupIBE**
  Initializes the system by generating its parameters and the master-key which will be used for the users private key generation.
- **ExtractIBE**
  Generates a private key based on the requesting users identity. Performed by the private key generator.
- **EncryptIBE**
  Encrypts the given message.
- **DecryptIBE**
  Decrypts the given ciphertext.

An IBS scheme is usually specified by four algorithms: {*SetupIBS*, *ExtractIBS*, *SignIBS*, *VerifyIBS*}, where

- **SetupIBS**
  Initializes the system by generating its parameters and the master-key which will be used for the users private key generation.

- **ExtractIBS**
  Generates a private key based on the requesting users identity. Performed by the private key generator.
- **SignIBS**
  Signs the given message.
- **VerifyIBS**
  Verifies the given signature.

4.2. *Formal Language Identity-based Encryption.* In this subsection, we provide a comprehensive description of FLIBE by presenting its algorithms. For a better understanding of the idea, we show how it works through a simple example and a flow chart of the system's operation.

There are three types of participants in the scheme. The sender (α) who encrypts a message, the receiver (β) who decrypts a cipher, and the private key generator (PKG).

- **Setup** Input: inherited from the used IBE and IBS schemes.
  Calls the *SetupIBE* and *SetupIBS* algorithms.
- **SignFormula** Input: An authorization formula Υ, the private key of the sender SK_α (which was generated with the *ExtractIBS* algorithm) and inheritance from the used IBS scheme.
  Generates the digital signature σ_Υ by calling the *SignIBS* function.
- **Evaluate** Input: An authorization formula Υ and the public key of the receiver PK_β.
  First, the algorithm checks for every attribute constraint in Υ, if PK_β satisfies it. So, if the current attribute constraint is $\Gamma = (N_\Gamma, L_\Gamma)$ and $\exists\ p \in PK_\beta$ property, such that $N_p = N_\Gamma$ and $V_p \cap L_\Gamma \neq \emptyset$, then PK_β satisfies Γ, else not. After the check is finished, the constraint is replaced with a boolean value in the formula and once all the checks are done, the last step is the evaluation of the authorization formula, which by now only contains boolean values and boolean operators.
- **GenerateEncryptionKey** Input: The public key of the sender PK_α.
  The function creates a property *gen*, where V_{gen} is a unique generated value. After that it appends *gen* to PK_α, resulting in a generated encryption key $PK_{\alpha_{gen}}$.
- **Extract** Input: An authorization formula Υ, a digital signature σ_Υ of it, the public key PK_α of the signature creator (the sender), the public key PK_β of the extractor (the receiver), the encryption key $PK_{\alpha_{gen}}$ and inheritance from the used IBE scheme.
  The PKG checks if Υ is authentic, with the verification of σ_Υ. After that it calls the *Evaluate* function. If it results with a positive output, the PKG generates the $SK_{\alpha_{gen}}$ private key by calling the *ExtractIBE* algorithm to the $PK_{\alpha_{gen}}$ encryption key, and outputs it to the extractor (β).

- **Encrypt** Input: An encryption key $PK_{\alpha_{gen}}$, a message M and inheritance from the used IBE scheme.

  Creates a ciphertext C by calling the *EncryptIBE* function.
- **Decrypt** Input: A decryption key $SK_{\alpha_{gen}}$, a ciphertext C and inheritance from the used IBE scheme.

  Calls the *DecryptIBE* function to achieve the decryption.

Figure 1 shows the full process of the scheme.

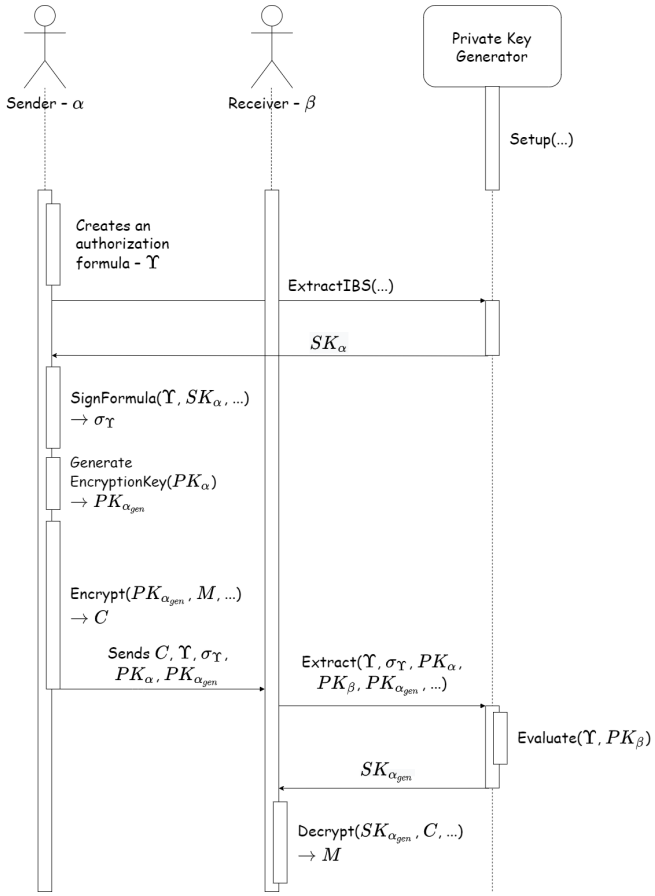

FIGURE 1. Formal Language Identity-based Encryption flow diagram

The scheme starts with an initialization function, called *Setup*. Its responsibility is to generate the public parameters and the master secret(s) of the used identity-based cryptography schemes.

The next step gives the essence of the scheme. Before the encryption, it is required to obtain an authorization formula. It is a logic formula that contains attribute constraints and boolean operations. The attribute constraint defines which are the required values for a given attribute, these combined with the logic operators, allows the definition of fine-grained access policy. In summary, with the authorization formula, the sender defines which entities are authorized to access the decryption key for the encrypted document.

Example 4.1.

("e-mail", .*@company\.com)
AND
("job", Lead developer, Chief Technology Officer)

As the example 4.1 shows, there is an authorization formula containing two attribute constraints combined with an AND operation. The first constraint is given with a regular expression and means an email address ending with *@company.com* is required. The second is given with an enumeration authorizing entities, with a property called job of which value equals one of the included values.

After the formula is created, the sender (α) creates a digital signature with a secure IBS. It is required to prevent the forge of the authorization formula.

In our scheme the public keys of the receivers are only used for authorization, the encryption is done with a generated key, which is the sender's public key (PK_α) expanded with a property with a unique value. This is in direct connection with the authorization formula, in the way that, if the formula did not change, there is no need to generate a new key, in this situation it is reusable. However, if the formula changed, it is of course required to generate a new key. Besides, the property values of the decryptors are constantly changing, which might be a problem, if one had access to the private key once, he/she can decrypt every cipher encrypted with the same key. In conclusion, it is recommended to operate with single-use encryption keys if the domain requires that. Also, since the encryption key is an expansion of PK_α, which is the verification key for the formula's digital signature, this guarantees for the PKG that the formula and the signature are created by α.

As Figure 1 shows after this step, the sender entity is ready to encrypt the message, with that finishing his part of the scheme.

Before the receiver could decrypt the ciphertext, it needs to do the *Extract* process to obtain the decryption key. Once it has finished, it can start the decryption.

Security. The security of our scheme is highly dependent on the used IBE and IBS schemes, so we assume they are secure. This means that the digital signature for the authorization formula Υ is correct, also the decryption key

$SK_{\alpha_{gen}}$ for the generated encryption key $PK_{\alpha_{gen}}$ cannot be guessed and the ciphertext is secure to the chosen ciphertext attack.

The *Evaluate* function should be secure, if Υ is not vulnerable, because, in the standard model, the adversary cannot extract a decryption key with a fitting public key. So we assume if the adversary tries to extract with any public key to the authorization formula Υ, the formula will not be satisfied.

Besides those, the only attackable surface is forging a new authorization formula. Since the used IBS scheme is secure, the adversary cannot replace the signed document without creating a new digital signature, also cannot extract and forge the digital signature of the original signer (α) for Υ. Furthermore, the adversary has no chance to replace Υ and its digital signature σ_Υ, because the encryption key $PK_{\alpha_{gen}}$ includes the signature verification key PK_α, so the deception would be obvious for the PKG.

Based on these, the scheme should be secure.

4.3. *Formal Language Identity-based Proxy Signature.* Fine-grained cryptographic access control is not restricted to encryption only, another obvious area is the proxy signature. With the definition of a policy, the users are able to automate the authorization method of proxy signers. It allows a higher level of unavailability for the user, which is one reason for the existence of this branch of cryptography, providing availability, through the proxy signers and not the original signer. Here we introduce our scheme for this purpose.

There are four types of participants in the scheme. The original signer (α) who authorizes proxies to sign documents on behalf of him, the proxy (β) who signs documents on behalf of α, the private key generator (PKG) and finally the verifier who checks the authenticity of the signature created by β.

- **Setup** Input: inherited from the used IBS scheme.
  Calls the *SetupIBS* algorithm.
- **SignFormula** Input: An authorization formula Υ and the private key of the original signer SK_α (which was generated with the *ExtractIBS* algorithm) and inheritance from the used IBS scheme.
  Generates the digital signature σ_Υ by calling the *SignIBS* function.
- **Evaluate** Input: An authorization formula Υ and the public key of the proxy PK_β.
  First, the algorithm checks for every attribute constraint in Υ, if PK_β satisfies it. So, if the current attribute constraint is $\Gamma = (N_\Gamma, L_\Gamma)$ and $\exists\ p \in PK_\beta$ property, such that $N_p = N_\Gamma$ and $V_p \cap L_\Gamma \neq \emptyset$, then PK_β satisfies Γ, else not. After the check is finished, the constraint is replaced with a boolean value in the formula and once all the checks are done, the last step is the evaluation of the authorization formula, which by now only contains boolean values and operators.

- **Extract** Input: An authorization formula Υ, a digital signature σ_Υ of it, the public key PK_α of the original signer, the public key PK_β of the proxy and inheritance from the used IBS scheme.

  The PKG checks if Υ is authentic, with the verification of σ_Υ. After that it calls the *Evaluate* function. If it results with a positive output, the PKG creates $PK_{\alpha\beta\Upsilon}$ with the concatenation of PK_α and PK_β and Υ and generates the private key $SK_{\alpha\beta\Upsilon}$ by calling the *ExtractIBS* function on it and outputs the private key.
- **Sign** Input: The private key $SK_{\alpha\beta\Upsilon}$, a message M and inheritance from the used IBS scheme.

  Creates a digital signature σ_M of M by calling the *SignIBS* function.
- **Verify** Input: An authorization formula Υ, a digital signature of the authorization formula σ_Υ, the public key of the original signer PK_α, a message M, a digital signature σ_M, the public key of the proxy PK_β and inheritance from the used IBS scheme.

  The verifier checks if Υ is authentic, with the verification of σ_Υ. After that it calls the *Evaluate* function. (If Υ contains appended warrant information and descriptions, the verifier checks that too.) If the checks were correct, the verifier inspects if σ_M is authentic by calling the *VerifyIBS* function.

Figure 2 shows the complete process of the scheme.

The flow of the FLIBPS and FLIBE are very similar. The only difference is the creation of the public key, which is part of the extraction function in the case of the FLIBPS.

With a slight modification on the authorization formula, it could fill the role of the warrant too, which would provide a partial delegation with a warrant proxy signature scheme.

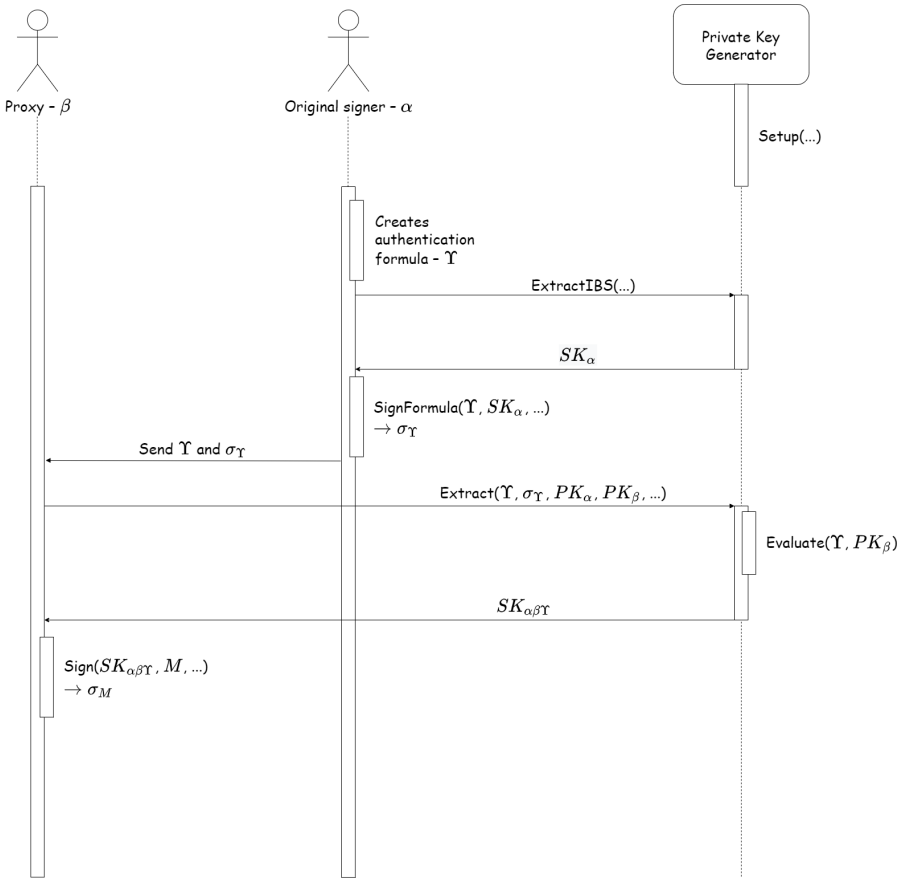

FIGURE 2. Formal Language Identity-based Proxy Signature flow diagram

5. PERFORMANCE

We created a proof of concept implementation of the Formal Language Identity-based Encryption scheme based on the CryptID library [24], which is available on GitHub[1]. It is a naive approach built on the Boneh-Franklin Identity-based Encryption [3] and the Hess Identity-based Signature [8], does not include any optimization or multi-thread implementation, however, it may give a better understanding of our concept.

[1] `https://github.com/cryptid-org/cryptid-native`

Besides the implementation of the scheme, we created benchmark tests too, which let us compare the performance of FLIBE and BSW-ABE [2], giving clear proof that FLIBE is more scalable. The benchmark was created with Google benchmark [6]. The measurement process included five repetitions, where every repetition consists of twenty iterations. The final results are the mean values of the runs.

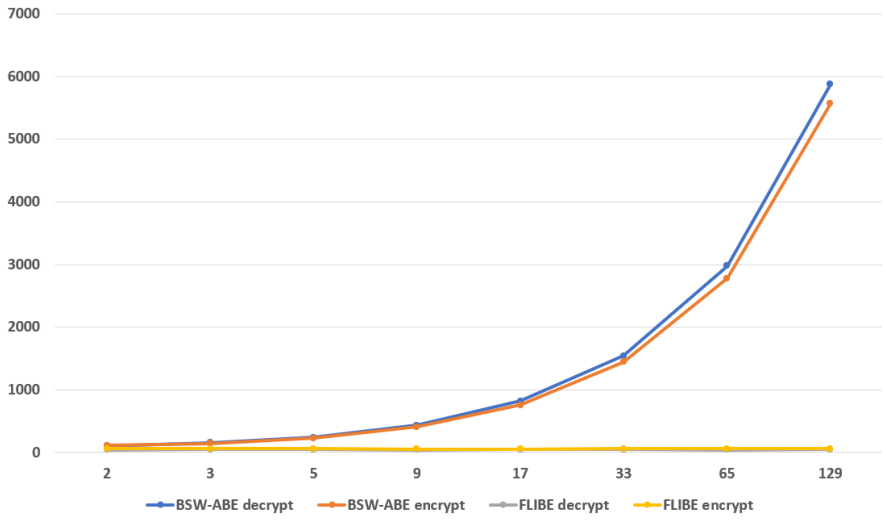

FIGURE 3. Performance comparison of FLIBE and BSW-ABE

In Figure 3 the x-axis represents the number of attributes included in the access policies and the y-axis is the runtime in milliseconds. The number of attributes is not knitted in the tested schemes, chosen like this for our convenience. The diagram shows that in the case of BSW-ABE the amount of computation is highly dependent on the number of attributes, also FLIBE is not affected by that, which makes the latter efficiently scalable and practical even in the case of complex access policies. As a remark, the FLIBE decrypt function has almost the same runtime as the encrypt function, this is why it is "hiding" on the diagram and only a little bit visible.

We also performed a test on FLIBE out of curiosity, where the number of attributes was set to millions. The result was that the performance was still not affected, stayed "constant".

These performance tests point out that we reached our goal of reducing the client-side computation, making the schemes practical, but it comes with a major drawback. The extract function, which is a server-side function used by the PKG. The minor addition to its calculations is the authorization formula verification and evaluation. The actual drawback comes from the fact that the

decryptors are using a private key created from a generated encryption key, which in practice cannot be a long-term key due to the continuous changes in the decryptor's property values (which is domain-specific). We think that the single-use encryption keys will be the most common use case. Therefore, the decryptors have to use the extract method regularly.

6. Conclusion

In this paper, we investigated the importance of fine-grained cryptographic access control today, because of the growth of the digital economy. We presented the current options and ideas for that kind of cryptographic schemes, with their benefits and drawbacks. We pointed out, that a general problem of these schemes is that the client-side computation is growing with the growth of the access policy, which makes most of them impractical.

After the examination of the current solutions, we presented our concepts, called Formal Language Identity-based Encryption and Formal Language Identity-based Proxy Signature. Our main focus was to eliminate the extra computation from the clients because in general, they have less computational power. We managed to achieve this with minimal growth on the server-side. The proof of concept implementation and benchmark tests are showing it clearly. Furthermore, we managed to reach it in a way, that our schemes are providing a more flexible access policy definition, with the use of formal language. These schemes seem to make fine-grained cryptographic access control practical.

Acknowledgements.

This work was partially supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was supported by the European Union, co-financed by the European Social Fund. The research was partially supported by the 2018-1.2.1-NKP-2018-00004 *Security Enhancing Technologies for the Internet of Things* project.

References

[1] M. Abdalla, D. Catalano, A. W. Dent, J. Malone-Lee, G. Neven and N. P. Smart, *Identity-based encryption gone wild*, in: Automata, Languages and Programming, Lecture Notes in Comput. Sci. **4052**, Springer, Berlin, 2006, pp. 300–311.

[2] J. Bethencourt, A. Sahai, and B. Waters, *Ciphertext-policy Attribute-based encryption*, in: 2007 IEEE Symposium on Security and Privacy (SP '07), IEEE, 2007.

[3] D. Boneh and M. K. Franklin, *Identity-based encryption from the Weil pairing*, in: Advances in Cryptology – CRYPTO 2001, Lecture Notes in Comput. Sci. **2139**, Springer, Berlin, 2001, pp. 213–229.

[4] F. Cai, N. Zhu, J. He, P. Mu, W. Li and Y. Yu, *Survey of access control models and technologies for cloud computing*, Cluster Computing **22** (2018), 6111–6122.

[5] C. Delerablée, *Identity-based broadcast encryption with constant size ciphertexts and private keys*, in: Advances in Cryptology – ASIACRYPT 2007, Lecture Notes in Comput. Sci. **4833**, Springer, Berlin, 2007, pp. 200–215.

[6] Google Benchmark – A microbenchmark support library, 2019. `https://github.com/google/benchmark`.

[7] V. Goyal, O. Pandey, A. Sahai and B. 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, ACM Press, 2006.

[8] F. Hess, *Efficient identity based signature schemes based on pairings*, in: Selected Areas in Cryptography, Lecture Notes in Comput. Sci. **2595**, Springer, Berlin, 2003, pp. 310–324.

[9] A. Huszti and N. Oláh, *A simple authentication scheme for clouds*, in: 2016 IEEE Conference on Communications and Network Security (CNS), IEEE, 2016.

[10] A. Huszti and N. Oláh, *Security analysis of a cloud authentication protocol using applied pi calculus*, International Journal of Internet Protocol Technology **12.1** (2019), pp. 16–25.

[11] I. Indu, P. R. Anand, and V. Bhaskar, *Identity and access management in cloud environment: Mechanisms and challenges*, Engineering Science and Technology, an International Journal **21.4** (2018), 574–588.

[12] S. Kim, S. Park and D. Won, *Proxy signatures, revisited*, in: Information and Communications Security, Lecture Notes in Comput. Sci. **1334**, Springer, Berlin, 1997, pp. 223–232.

[13] M. Mambo, K. Usuda and E. Okamoto, *Proxy signatures: Delegation of the power to sign messages*, IEICE Trans. Fundamentals, A **79.9** (1996), 1338–1354.

[14] T. Okamoto and K. Takashima, *Fully secure unbounded inner-product and Attribute-based encryption*, in: Advances in Cryptology – ASIACRYPT 2012, Lecture Notes in Comput. Sci. **7658**, Springer, Heidelberg, 2012, pp. 349–366.

[15] R. Ostrovsky, A. Sahai and B. Waters, *Attribute-based encryption with non-monotonic access structures*, in: Proceedings of the 14th ACM conference on Computer and communications security – CCS '07, ACM Press, 2007.

[16] R. Pompon, *Is the Cloud Safe? Part 2: Breach Highlights for the Past 3 Years*, Dec. 2019. `https://www.f5.com/labs/articles/threat-intelligence/is-the-cloud-safe--part-2--breach-highlights-for-the-past-3-years`.

[17] A. Sahai and B. Waters, *Fuzzy Identity-based encryption*, in: Advances in Cryptology – EUROCRYPT 2005, Lecture Notes in Comput. Sci. **3494**, Springer, Berlin, 2005, pp. 457–473.

[18] R. Sakai and J. Furukawa, *Identity-based broadcast encryption*, IACR Cryptol. ePrint Arch. 2007:217, 2007.

[19] A. Salomaa, Formal languages, Academic Press, New York, 1973.

[20] A. Shamir, *Identity-based cryptosystems and signature schemes*, in: Advances in Cryptology – CRYPTO 84, Lecture Notes in Comput. Sci. **196**, Springer, Berlin, 1985, pp. 47–53.
[21] C. C. Tan, H. Wang, S. Zhong and Q. Li, *IBE-Lite: A lightweight Identity-based cryptography for body sensor networks*, IEEE Transactions on Information Technology in Biomedicine **13.6** (2009), 926–932.
[22] J. Tomida, Y. Kawahara and R. Nishimaki, *Fast, compact, and expressive Attribute-based encryption*, in: Public-key cryptography – PKC 2020. Part I, Lecture Notes in Comput. Sci. **12110**, Springer, Cham, 2020, pp. 3–33.
[23] Using MIKEY-SAKKE: Building secure multimedia services, `https://www.ncsc.gov.uk/whitepaper/using-mikey-sakke--building-secure-multimedia-services`.
[24] Á. Vécsi, A. Bagossy and A. Pethő, *Cross-platform Identity-based cryptography using WebAssembly*, Infocommunications journal **11.4** (2019), pp. 31–38.
[25] L. Zhang, Y. Hu and Q. Wu, *Adaptively secure identity-based broadcast encryption with constant size private keys and ciphertexts from the subgroups*, Math. Comput. Modelling **55** (2012), pp. 12–18.

Kriptografija formalnog jezika temeljena na identitetu

Ádám Vécsi i Attila Pethő

Sažetak. Brzi rast digitalne ekonomije čini kontrolu pristupa sve izazovnijom. Jedno od najviše pogođenih područja je računalstvo u oblaku, koje iz sigurnosnih razloga zahtijeva kriptografsku kontrolu pristupa. Trenutno je najbolje rješenje za to upotreba kriptografije temeljene na atributima, koja omogućuje definiranje pravila pristupa, na temelju atributa entiteta. Nažalost, ova familija shema ima značajan nedostatak, budući da potrebno računanje od strane korisnika raste s rastom složenosti kontrole pristupa. U ovom radu, dajemo koncept, nazvan kriptografija formalnog jezika temeljena na identitetu, koji daje rješenje ovog problema, čineći kriptografsku kontrolu pristupa praktičnom.

Ádám Vécsi
Department of Computer Science
University of Debrecen
Kassai str. 26
H-4028 Debrecen, Hungary
E-mail: `vecsi.adam@inf.unideb.hu`

Attila Pethő
Department of Computer Science
University of Debrecen
Kassai str. 26
H-4028 Debrecen, Hungary
E-mail: `Petho.Attila@inf.unideb.hu`

Received: `31.10.2020.`
Revised: `11.5.2021.`
Accepted: `18.5.2021.`