Unlinkable Attribute-Based Credentials

Research seminar in Cryptography

Research paper

Supervisor:	Ahto Truu
Student:	Kristiina Konno B02315
E-mail:	Kristiina.konno@taltech.ee
Study programme:	IVCM - cybersecurity

Tallinn 2019

Table of Contents

Introduction 3
1.What are attribute-based credentials 3
2.Background information 5
2.1.Zero-knowledge proofs 5
2.2.Pseudorandom Number Generators 5
3.Idemix 6
3.1.Overview of Idemix 6
3.2.Camenisch-Lysyanskaya signature scheme 7
3.3.Attributes and credentials 8
4.IRMA 8
5.U-Prove 10
6.Comparison between Idemix, IRMA and U-Prove 10
7.Conclusion 11
References 12

Introduction

We have to remember and use countless different attributes in every day and are also used to giving redundant information to prove different attributes about ourselves. Attribute-based credentials only reveal the minimum needed information about a user, thus aiding privacy. There have been several different systems realizing the concept, the main ones being Idemix, U-prove, and IRMA. In this paper, we will take a closer look at them, to see what their differences are, and whether and how they can be implemented on smartcards..

1.What are attribute-based credentials

By now, we are all used to having dozens of different accounts and having to remember all the different passwords. Many people also tend to use, for example, Google or Facebook as a login to several different places. With that, we reveal to services like Facebook and Google too much information about our online activities and thus compromising privacy unnecessarily. We also tell the other services too much about ourselves. Let's say we use Google to log into a news site as well. Not only does Google have the information that we read the news there, this news site asked us to share some of our information from Google. If we want to use the site, we have to agree to share that information. And the problem is that more often than not, the site asks for too much information. [1]

Attribute-based credentials allow us to take back control over our privacy. They allow users only to send over the attributes that are needed. There are a lot of different attributes that describe us – name, gender, date of birth, and many others. All the different attributes are gathered into cryptographic containers, which are then signed. The user can then choose which attributes to disclose, and the validity of those attributes can still be verified. Users can also stay anonymous if the proofs are unlinkable. [1] There are three different kinds of attributes: values that the issuer knows, has a commitment to, or some values completely hidden from the issuer. The value being hidden from the issuer implies the issuer does not have access to it during the issuing phase. For example, let us say that the government is issuing the identity cards that have an attribute-based credential system run on it. Then, if they are issuing cards to citizens, they are also issuing the master secret stored on it, but they do not know the value of the master secret. In this example, the name will be value the issuer and the user are aware of. Zero-knowledge proofs are used to verify hidden types so that the contents of the hidden attribute are never revealed in the verification process. The hidden type comes into play when talking about IRMA in later sections, and we will explain it in detail in the next chapters. [2]

An issuer knows the commitment to an attribute is the equivalent of the user sealing the attribute into an envelope, and giving the issuer the envelope. The main benefits of a commitment to a message are the correctness of the message as well as the hiding and binding properties. Meaning, the commitment of a message by the user does not give the issuer any information about the message. At the same time, no two messages commit to the same value, and when the user reveals the message, it can be easily proven that the message did result in the given commitment. [3] The difference between the hidden attributes and the committed attributes is that the original message of the committed attribute can be later revealed, but hidden attributes remain hidden indefinitely.

Whether it is an email, personal ID code, or something different, we all have something unique that identifies us that we use to log into systems. However, these labels create a privacy risk. For example, in Tallinn, residents have had free transport for a while now. And it is all connected to each subject's ID card number. There were some considerable security risks in the beginning [4], and when all that data was connected to the users' personal code, security was compromised. The problems were that the server holding the data was not secure enough, and it contained all the information of a person's travels. Because people were identified by their personal code, then it was possible to connect a person to their travels, and that created a considerable risk to people's privacy. In reality, the system only needs to know whether you have the right to use public transport. [5]

Another use case would be shopping in different stores. For example, in order to buy alcohol, only the fact that one is of legal drinking age is needed, but not the name and full date of birth. Attribute-based credentials allow us to achieve exactly that. Another example would be in anonymous chat rooms that have an age limit. Attribute-based credentials give the option to remain anonymous, but at the same time, prove that one is of the right age. [6]

2.Background information

Here we explain some background information needed to understand credential schemes.

2.1.Zero-knowledge proofs

The main goal of zero-knowledge proofs is to verify data without revealing the data. There are always two parties involved – the prover and the verifier. The prover holds the sensitive data that the verifier wants to verify. In order to do that, an indirect proof is created, and if that holds, then the data is verified. [7] Let's look at a simple example. Let's say that Alice and Bob are playing a game of 'where is, Waldo?'. Alice wants to prove to Bob that she has found Waldo without revealing his location. One way to do that is that Alice cuts out Waldo's shape from a big piece of paper. Alice can then overlay the paper over the image where waldo is. This way, Bob does not find out the location of Waldo, but he can see that Alice has found it. And the probability that Alice cut out a random shape and it matched the location of Waldo is minuscule. Thus, we have found a way for Alice to prove that she has found Waldo, but Bob will not know the exact location. [8]

2.2. Pseudorandom Number Generators

We are all familiar with the concept of random numbers, meaning a sequence of uniformly distributed numbers where it is not possible to foresee upcoming numbers based on previous ones. But that is, in fact, very complicated to achieve in computers. Truly random numbers exist in real life. They exist in physical phenomena, for example, in atmospheric noise, quantum phenomena, and other natural phenomena. But the problem is that the computer is very deterministic and thus is incapable of producing the indeterministic sequence. Computers, however, have a way to generate pseudorandom numbers. Meaning, a sequence of numbers that is close to a truly random number. They do that by taking a small seed of a truly random number from the noise coming from mouse events or audio. This seed is used as an input for a pseudorandom number generator, which then performs several mathematical functions on the seed, and produces a list of pseudorandom numbers. This randomness of the final sequence is dependent on the randomness of the initial seed. There is a period in which the sequence will start repeating itself. Thus the length of the pseudorandom number sequence must be strictly less than the period. The period is dependent on the length of the seed and the algorithm itself. In order for a pseudorandom sequence and a truly random sequence to be truly indistinguishable from each other, it must be unrealistic for a computer to look for a match by generating a pseudorandom sequence with all possible seeds in a reasonable amount of time. [9]

3.Idemix

The next step in understanding attribute-based credentials is looking at the most popular systems that have implemented it. We will start by looking at the identity mixer cryptography library, or Idemix for short.[1] To start, we will take an overall look at the library, then take a deeper look into the Camenisch-Lysyanskaya (CL) signature scheme that Idemix uses and conclude by discussing attributes and credentials in more detail.

3.1.Overview of Idemix

Identity Mixer Anonymous Credential System, or Idemix, is actually based on anonymous credentials, but has led the path for attribute-based credentials. In an anonymous credential system, there are issuers who issues the credentials, and verifiers who receives them. The credential is made up of a set of attributes as well as information needed for the proof. For other tasks such as credential revocation, anonymity revocation and other, a trusted third party is used. Each user also has their master secret which is encoded into every credential. The master secret is never revealed in the credential issuing phase. In this paper, we will not go into pseudonyms, but the master secret is used to derive a non-deterministic pseudonym, and the master secret is its password. One master secret can produce multiple pseudonyms.. [2]

There are two separate protocols in Idemix – credential issuance and credential proof. It starts with the user requesting a credential from the issuer. After that, the issuer and user agree upon which attribute values to include in the credential and as well as the credential structure. The user then runs the credential issuing protocol together with the issuer, which is, in reality, a CL signature scheme that signs the hash of the message containing the attribute with the issuers public- and secret key. After that, the user can prove to a verifier the possession of the credential. During the verification process, the user does not send over the actual signature, but just the randomized values of the signature. The user then proves that she knows the corresponding signature. [2]

3.2.Camenisch-Lysyanskaya signature scheme

As we mentioned, then Idemix is built upon the Camenisch-Lysyanskaya (CL) signature scheme. This scheme can be used to sign blocks of messages with a single signature. Each attribute value can be considered as a separate message, or they can be combined into one message. For committed attributes, the commitment of the attribute value is used as the message. What makes this signature scheme so unique, is that it allows a user to prove possession of a signature using efficient zero-knowledge proofs of knowledge. [2]

The signature scheme has global parameters $Z, S, R \in QR_n$ where QR_n is the multiplicative subgroup of quadratic residues $QR_n \leq Z_N^*$. A user's private key is (p, q) where p and q are large primes and the public key is n = pq. [2]

The CL signature on message m is (A, e, v), where v is a random integer, e is a random prime and $A \equiv \left(\frac{Z}{S^v R^m}\right)^d (mod n)$, where $d \bullet e \equiv 1(mod \varphi(n))$ and

$\varphi(n) = (p-1)(q-1)$. In case of multiple attributes the messages are $m_0, m_1, \dots, m_l$ and $A \equiv \left(\frac{Z}{S^v \prod_{i=0}^{l} R_i^{m_i}} \right)^d (mod n)$. [2]

The verification of a single message $Z \equiv A^e S^v R^m (mod n)$ and in the case of multiple messages $Z \equiv A^e S^v \prod_{i=0}^{l} R_i^{m_i} (mod n)$ [2]

The security of the scheme comes from the so-called RSA assumption that large integers are too difficult for computers to factor, thus it is infeasible that someone will find out p and q from just knowing n. [2]

3.3.Attributes and credentials

There are several components in the Idemix library, the central one being the attributes. Each attribute is a triple consisting of name, type and value. Idemix supports data types such as strings, integers, dates and enumerations. A credential is a set of attributes that can be distinguished as 3 different kinds: attributes whose value the issuer knows, that have a committed value, or where the value is completely hidden from the issuer. [2]

The two latter kinds are implemented via commitments. We denote a value v committed by a user as $C \leftarrow Comm(v)$. The commitment has two properties – a hiding and a binding property. Hiding refers to the recipient not being able to infer information about v given C and binding refers to the committer not being able to convince a recipient that $C = Comm(v')$ for a $v' \neq v$ [2].

4.IRMA

The next attribute-based credential system that we will take a closer look at is "I Reveal My Attributes" or IRMA. It is a realistic implementation of the Idemix system. IRMA does not implement all possible features of Idemix [1]. They use a simplified version of Idemix that only has the basic features. In practice, this does not restrict users. However, some features, like domain-specific pseudonyms, are not implemented. IRMA has been implemented as a smartphone application, as well as a smartcard. The main reason for implementing just a simplified version on a smartcard insufficient the processing power of the smartcards to run full Idemix. One of the problems was that implementing the Idemix with all of the possibilities would take anywhere from 4 to 10 seconds to just prove one attribute. The goal of the IRMA card was to implement only the necessary parts and cut the runtime down to 1-1.5 seconds. The achieve that, IRMA restricted the size of the attributes and other parameters. From there, we can calculate the latency of the IRMA implementation. The verification time could be further optimized by recomputing the pseudorandomness that is used in the attributes and reducing the overhead of the computations for hiding the attributes. IRMA saves the signatures and credential attributes in the flash memory. The intermediate values and the randomized signature are computed in the RAM, but since it has only that much memory space, then the card allows five values and the master secret. The main difference between the flash memory and RAM is that even though the latter is lost when the power is lost, then writing to flash memory is very slow. [10]

One place to make the current implementation better is to rearrange the memory storage so that we could generate as many intermediary values as possible. The intermediary values are used during the computation of the commitment and for hiding the desired attributes. The values are pseudorandom, used in hiding an attribute, or in calculating its commitment, which is later used in the credential proof. There have been several suggestions on how to improve on that. One of which is to use PRNGs to regenerate the random exponent, but none of them gave any tangible benefit to the delay. [10]

One option is to replace the intermediary values with an AES key to generate the intermediary values as needed. This is used in two different places, and overall reduces the usage of RAM by around 300 bytes. But extra latency is added when computing all of the intermediary values and a totally new latency when computing the pseudorandomness at each value. So, we won 300 bytes of RAM at the cost of adding 40 ms to 200 ms of latency, depending on how many attributes are hidden. [10]

5.U-Prove

U-Prove was developed back in 2000 and was later acquired by Microsoft.[11] This technology revolves around the U-Prove token which is a collection of attributes. In order to issue a credential, the issuer and the user must share, among other values, the attribute, and then the attribute is digitally signed. The verifier can then verify the signature with the token presentation protocol. In this protocol, the user gives to the verifier the attribute, the public signature and the token-specific public key. The user also adds a response, which is a signed pseudorandom number and the timestamp. This proves that the used hold the correct private key without revealing it. [12]

The token holds several attributes that can either be hidden or shown depending on how to prover chooses.[11] This token also includes the token information field and the prover information field. The token information field includes metadata, such as the usage restrictions and validity period, that are encoded and always disclosed to verifiers. The prover information field is used to hide information, such as the encryption key or nonce used by the verifier, from the issuer. Every token also comes with the issuers signature on all the token contents. [12]

6.Comparison between Idemix, IRMA and U-Prove

We now have a brief overview of the main attribute-based credential systems. Idemix has a very strong and solid cryptographic base, but building a usable app upon that is rather difficult because, even modern smartcards aren't powerful enough to run Idemix with its full opportunities. To be more precise, it is possible to run Idemix fully on a smartcard, but verification and other processes would take too long to run, thus making it infeasible. Irma wanted to change that and made Idemix run with a feasible time on a smartcard. And they achieved a runtime of 1-1.5 seconds, but the downside is that they cut down the number of available attributes. U-prove takes another route with its attribute-based credentials and uses tokens. With it, it is possible to have the same feasible runtime on smartcards. Thus, there is a case to be made, that U-Prove's approach is better when applying it to smartcards.

Idemix has several features, that are infeasible to run on smartcards, but this problem might be solved, by implementing idemix on smartphones. The main benefits that Idemix has on U-Prove are the pseudonyms that allows the user to stay secret by hiding their real name. That brings more privacy to the users while at the same time they are able to verify attributes about themselves.

7.Conclusion

We have taken a look at the different attribute-based credential systems. There are several benefits to the users privacy in using these credential systems, because with these systems people are not forced to give out redundant information. These systems do come with their setback. The main one being actually running them on smartcards, because smartcards don’t have the capabilities to run an attribute-based credential system with their full theoretical possibilities.

References

1. Brinda Hampiholi, Gergely Alpár, Fabian van den Broek, Bart Jacobs, Wouter Lueks, and Sietse Ringers. *IRMA: practical, decentralized and privacy-friendly identity management using smartphones*. 10th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2017), Minneapolis, USA, 2017.

2. “Specification of the Identity Mixer Cryptographic Library (Revised version 2.3.0)*,” *IBM Research | Technical Paper Search | Specification of the Identity Mixer Cryptographic Library (Revised version 2.3.0)*(Search Reports)*, 14-Sep-2016. [Online]. Available: https://domino.research.ibm.com/library/cyberdig.nsf/1e4115aea78b6e7c85256b360066f0d4/eeb54ff3b91c1d648525759b004fbbb1?OpenDocument. [Accessed: 25-Nov-2019].

3. K. Rannenberg, J. Camenisch, and A. Sabouri, *Attribute-based Credentials for Trust Identity in the Information Society*. Cham: Springer International Publishing, 2015.

4. “Ühiskaardi süsteemi vead püütakse maha vaikida,” *Eesti Päevaleht*, 01-Jan-6206. [Online]. Available: https://epl.delfi.ee/eesti/uhiskaardi-susteemi-vead-puutakse-maha-vaikida?id=65547608. [Accessed: 25-Nov-2019].

5. Lejla Batina, Jaap-Henk Hoepman, Bart Jacobs, Wojciech Mostowski, and Pim Vullers. *Developing efficient blinded attribute certificates on smart cards via pairings*. In: D. Gollmann and J.-L. Lanet, editors, Smart Card Research and Advanced Applications, 9th IFIP WG 8.8/11.2 International Conference, CARDIS 2010, Springer LNCS 6035, p.209-222.

6. IRMA_privacy, *Privacy by Design Foundation*. [Online]. Available: https://privacybydesign.foundation/irma-explanation/. [Accessed: 25-Nov-2019].

7. C. Rackoff and D. R. Simon, “Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack,” *Advances in Cryptology — CRYPTO ’91 Lecture Notes in Computer Science*, pp. 433–444.

8. N. Zhu, “Understanding Zero-knowledge proofs through simple examples,” *Medium*, 12-May-2019. [Online]. Available: https://blog.goodaudience.com/understanding-zero-knowledge-proofs-through-simple-examples-df673f796d99. [Accessed: 19-Dec-2019].

9. “Pseudorandom number generators,” *Khan Academy*. [Online]. Available: https://www.khanacademy.org/computing/computer-science/cryptography/crypt/v/random-vs-pseudorandom-number-generators. [Accessed: 19-Dec-2019].

10. A. D. L. Piedra, J.-H. Hoepman, and P. Vullers, “Towards a Full-Featured Implementation of Attribute Based Credentials on Smart Cards,” *Cryptology and Network Security Lecture Notes in Computer Science*, pp. 270–289, 2014.

11. W. Mostowski and P. Vullers, “Efficient U-Prove Implementation for Anonymous Credentials on Smart Cards,” *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering Security and Privacy in Communication Networks*, pp. 243–260, 2012.

12. C.Paquin and G. Zaverucha, “U-prove cryptographic specification v1. 1 (revision 3)(December 2013).” *Microsoft.*