Data Is Key: Introducing the Data-Based Access Control Paradigm

Wolter Pieters and Qiang Tang

Faulty of Electrical Engineering, Mathematics and Computer Science
University of Twente
P.O. Box 217, 7500 AE ENSCHEDE, The Netherlands
{w.pieters,q.tang}@utwente.nl

Abstract. According to the Jericho forum, the trend in information security is moving the security perimeter as close to the data as possible. In this context, we suggest the idea of data-based access control, where decryption of data is made possible by knowing enough of the data. Trust is thus based on what someone already knows. A specific problem is defined as follows: given n pieces of data, an agent is able to recover all n items once she knows k of them. The problem is similar to both secure sketches and secret sharing, and we show that both can be used as a basis for constructions. Examples of possible applications are granting access without credentials, recovering forgotten passwords and sharing personal data in social networks.

Keywords: data-based access control, de-perimeterisation, secure sketches, secret sharing.

1 Introducing Data-Based Access Control

1.1 Motivation

According to the Jericho forum [4], traditional boundaries in information security are disappearing. Organisations are flexible and outsource part of their processes, employees work from home and mobile devices contain loads of data. Instead of relying on firewalls, the protection should therefore lie as close to the data as possible. In this paper, we address this issue from the perspective of access control.

In general, mainstream access control relies on two separate mechanisms: authentication and authorisation. It is first determined if the user has the claimed identity, and based on this identity certain privileges or trust are assigned. In a de-perimeterised setting, however, we may not be so sure anymore about the identity of persons, or the current privileges that should be coupled to it. Because people join and leave organisations on a regular basis, information about their identity may well be outdated. Especially when roles of persons change often and are not administered timely, identity can be an unreliable intermediary. In this context, the question can be asked whether the two steps can be merged into a single one. Can we trust people without knowing their identity?

Often, authentication and authorisation of persons is based on what a person knows. Normally, this is a username-password combination or a key. We radicalise this idea here, and focus on whether it is possible to let *any* piece of data serve as part of the

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 240–251, 2009.

access control policy. More specifically, we concentrate on access control policies in which the data protects itself. That is, an agent acquires access to new data based on the data it already possesses. We call this approach *data-based access control*. In the data-based access control approach, identity is no longer the focal point of authorisations. Instead, authorisations are coupled directly to credentials. These credentials can be any form of data: passwords, keys, files, or other pieces of data.

As an example, consider an electronic health record of a patient. We assume here that the file resides in a protected form on a public server. Either the authorities or the patient herself will have to administer the access policies for the file. How can they set the access control such that a practitioner can update her copy of the file to a new version, without first having to authenticate herself? Most importantly, how can we assign the access rights for the current file without relying on authentication? Basically, practitioners that have been treating the patient before will possess an earlier version of the medical file, including part of the information of the new version. This means that the patient can grant access to the updated version based on the earlier version as a credential. Apart from practical challenges in managing such access rights, the key feature is that these rights are based on data rather than special credentials.

If the patient does not trust the possessors of the previous version anymore, she may set a different access policy for the new file, in which an additional credential is required. She may then distribute this credential to the trusted practitioners via a different channel. How such policies can be implemented will be discussed in the following. The main point is that it is possible to base access control on information rather than identity.

In this paper, we define the characteristics of data-based access control and the associated policies as a new framework for data protection. We also show that existing cryptographic primitives can be used to implement the policies. The main contribution of this paper is therefore the definition of a new research area and its connection with already existing cryptographic primitives.

1.2 Problem Statement

Data-based access control is a form of access control where the possibility of decryption is protected by the data itself, not by separate credentials such as keys or attributes. Data-based access control is based on *similarity* between what the user wants to know and what she already knows. The central assumption is that if one already knows much within a specific domain, one has the right to know more.

Definition 1. Data-based access control *is the granting of access to information based on the similarity between the information being accessed and the information provided to access it.*

In this paper, we introduce (k, n) threshold data-based access control. More precisely, if from a set of n data-items the agent already knows k, she can recover the remaining items. Also, when the agent knows fewer than k, the uncertainty about the remaining items is equal to or only negligibly smaller than the entropy in each data item.

1.3 Applications

Data-based access control may be used for various purposes:

- authorisation without identification, where trust is based on what an agent already knows;
- recovering forgotten passwords; if an agent uses n passwords, she may store these in such a way that she can recover one if she remembers the other $n - 1$;
- sharing data in social networks; people may want to reveal more to people who already know a lot about them;
- secure version management; access can be granted to a new version of a file based on knowledge of earlier versions.

1.4 Related Work

Developments in the direction of data-level security include sticky policies [6] and attribute-based encryption [8]. The sticky policies paradigm involves the idea that policies can be associated to data in such a way that they are enforced by the association mechanism itself. Attribute-based encryption is such a mechanism.

In *attribute-based encryption* [8], the possibility of decryption is dependent on the receiver's attributes rather than her knowledge. These attributes have been verified by a key authority, which has issued a key corresponding to these attributes. In data-based access control, there is no independent key authority, since possession of part of the data immediately proves that one is entitled to know the rest.

An idea similar to data-based access control is the *fuzzy vault* [5]. There, a secret value is "locked" by a set of public values. If a sufficiently close set of public values is input, the vault can be unlocked. In a fuzzy vault, however, the elements for unlocking the vault are from a publicly announced domain, which is different from the secret we wish to protect. This is not the case in data-based access control: here, the elements for unlocking are themselves also the secrets.

Data-based access control can be conceived as a *secure sketch* problem. A secure sketch [1] is a procedure that maps a secret value to a public bit string, where the public bit string plus an approximation of the secret value allow for recovery of the secret value, provided the approximation is close enough. Both problems involve the publication of information that allows one to reconstruct the original data if one can approximate the original data closely enough. Indeed, the information published in data-based access control can be thought of as a secure sketch, namely one in which the distance measure is the set difference between the stored data-items and the remembered data-items. However, secure sketches have been mainly applied to problems where the input data is inherently fuzzy (like biometrics), where characteristics of the biometric template can be described in terms of feature sets. In this paper, we focus on the (k, n) threshold data recovery problem instead, where it may or may not be allowed to approximate some or all of the individual items. Also, as opposed to secure sketches, data-based access control is not focused on error correction. For decryption, an agent should input exactly k known values, not n values of which at least k are correct.

An approach similar to data-based access control was suggested for authentication purposes [3]. In this "personal entropy" scheme, an authentication secret is divided using Shamir's secret sharing [9]. For each share, a question together with the correct answer allows one to decrypt the share of the secret. The scheme that is proposed may also be applied in data-based access control. In the current work, however, the essential feature is that it is the data itself that is being protected, not an authentication key.

Recently, it was proposed to use digital objects as passwords [7]. The objects used in that framework are public information such as music or pictures rather than protected information. Also, the approach is again focused on acquiring a key rather than accessing the protected data directly.

1.5 Outline of the Paper

In section 2, we introduce the principles of data-based access control. In section 3, we describe a possible solution for the problem of (k, n) threshold data-based access control based on polynomial interpolation. In section 4, we show how an order-invariant scheme can be devised using secure sketches. In section 5, we discuss our prototype system. In section 6, we provide some suggestions for asymmetric schemes. In section 7, we evaluate the approach and solution. In section 8, we conclude and suggest further research.

2 Principles

In the traditional approach of access control, there are two phases: authentication and authorisation. Authentication specifies how the possession of certain credentials establishes an identity. Authorisation specifies how a certain identity leads to access.

Data-based access control does not distinguish a priori between credentials and data. It assumes a distribution over the participating agents of data-items from the set that we are interested in ($\mathcal{U}$) and defines how access to these items is related to access to other items. In the notation, we use $\mathcal{D}$ for sets of data items and D for individual data items.

We can distinguish between non-interactive and interactive forms of data-based access control. In interactive forms, agents can execute a protocol, which in the end provides the desired distribution of data (figure 1). In non-interactive forms, an agent publishes its data on a public bulletin board with the policies "attached", after which other agents with the appropriate possessions can access it. We focus on non-interactive forms here.

A data-based access control policy consists of a domain $\mathcal{U}$ and a function $\mathcal{P}(\mathcal{U}) \rightarrow \mathcal{P}(\mathcal{U})$, mapping sets of required items to sets of items that the required items give access to ($\mathcal{P}$ denotes power set). A policy is well-formed if a) data-items give access to themselves, and b) more data-items give access to more data-items.

Definition 2. *A data-based access control policy is* well-formed *if for each mapping* $\mathcal{D}_1 \mapsto \mathcal{D}_2$*:*

- $\mathcal{D}_1 \subseteq \mathcal{D}_2$*, and*
- *for each data item* D *and mapping* $\mathcal{D}_1 \cup \{D\} \mapsto \mathcal{D}_3$*:* $\mathcal{D}_2 \subseteq \mathcal{D}_3$.

The intuition here is that one already has access to the data-items one supplies as credentials, so a policy denying this access would not be enforceable. Moreover, if a set of data-items does not give one access to a specific piece of data, but a subset does, one can use the subset instead and still get the piece of data. Thus, again, such a policy would not be enforceable.

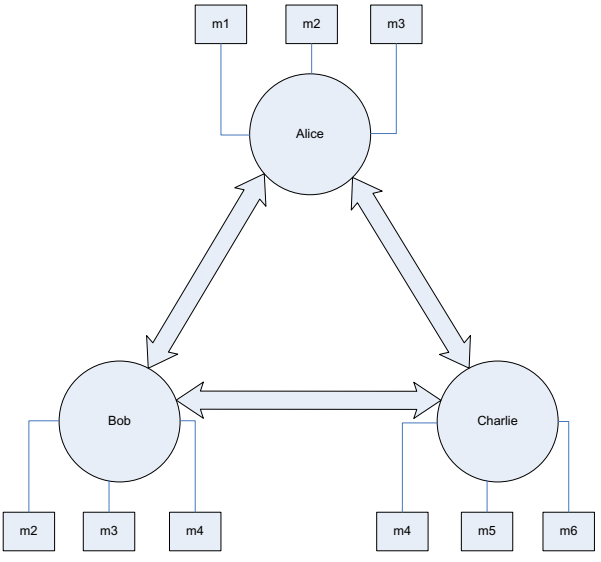

Fig. 1. Interactive data-based access control. Alice, Bob and Charlie each possess a set of messages. They wish to share all of their messages with another person if they already share at least two.

Example 1. We give an example of a well-formed policy:

- $\emptyset \mapsto \emptyset$
- $\{D_1\} \mapsto \{D_1\}$
- $\{D_2\} \mapsto \{D_2\}$
- $\{D_3\} \mapsto \{D_3\}$
- $\{D_1, D_2\} \mapsto \{D_1, D_2, D_3\}$
- $\{D_1, D_3\} \mapsto \{D_1, D_2, D_3\}$
- $\{D_2, D_3\} \mapsto \{D_2, D_3\}$
- $\{D_1, D_2, D_3\} \mapsto \{D_1, D_2, D_3\}$

We show that the fourth line satisfies the requirements. Trivially, $\{D_3\}$ is a subset of itself (first requirement). Also, adding a data item gives the mappings $\{D_1, D_3\} \mapsto \{D_1, D_2, D_3\}$ and $\{D_2, D_3\} \mapsto \{D_2, D_3\}$. In both cases, $\{D_3\}$ is a subset of the image (second requirement). The requirements can similarly be verified for the other lines.

We distinguish between *symmetric* and *asymmetric* data-based access control. Symmetric access control can only be used with symmetric policies. A symmetric policy is a well-formed policy in which the items in the rules are equivalent: if I can get a by knowing b, then I can also get b by knowing a. More precisely, if I can get any *additional* item D based on a set of known items $\{D_1 ... D_n\}$, I can also retrieve any item from this set if I do not know that item, but I do know D.

Definition 3. *A data-based access control policy is* symmetric *if it is well-formed, and for each mapping $\mathcal{D}_1 \mapsto \mathcal{D}_2$, for which $D_1 \in \mathcal{D}_1$, $D_2 \in \mathcal{D}_2$ and $D_2 \notin \mathcal{D}_1$, there also exists a mapping $(\mathcal{D}_1 \setminus \{D_1\}) \cup \{D_2\} \mapsto \mathcal{D}_3$, where $D_1 \in \mathcal{D}_3$.*

Example 2. The policy of the previous example is not symmetric. With D_1 and D_2, one can get D_3 (fifth line), but when reversing D_1 and D_3, one cannot get D_1 with D_2 and D_3 (seventh line).

Symmetric data-based access control is the focus in the current work; some ideas on asymmetry are discussed in section 6. More specifically, we focus on (k, n) threshold data-based access control.

Definition 4. *A data-based access control policy for universe $\mathcal{U}$ is a* (k, n) threshold policy *if:*

- $|\mathcal{U}| = n$,
- *for each $\mathcal{D} \subseteq \mathcal{U}$ with $|\mathcal{D}| \geq k$, $\mathcal{D} \mapsto \mathcal{U}$, and*
- *for each $\mathcal{D} \subseteq \mathcal{U}$ with $|\mathcal{D}| < k$, $\mathcal{D} \mapsto \mathcal{D}$.*

Lemma 1. *A (k, n) threshold policy is symmetric.*

Proof. We first prove that a (k, n) threshold policy is well-formed. Because $\mathcal{D}$ either maps to $\mathcal{U}$ or to itself, the first requirement of well-formedness is satisfied, since $\mathcal{D}$ is both a subset of itself and of $\mathcal{U}$. Furthermore, in the first case, $|\mathcal{D}| \geq k$ and therefore $|\mathcal{D} \cup \{D\}| \geq k$. $\mathcal{D} \cup \{D\}$ will thus also map to $\mathcal{U}$, which is a subset of itself, proving the second requirement. In the second case, $\mathcal{D} \cup \{D\}$ will map to either $\mathcal{D} \cup \{D\}$ or $\mathcal{U}$, of both of which $\mathcal{D}$ is a subset.

To prove that a (k, n) threshold policy is also symmetric, we look at the policy rule $\mathcal{D}_1 \mapsto \mathcal{D}_2$ and distinguish again the cases $|\mathcal{D}_1| \geq k$ and $|\mathcal{D}_1| < k$. In the first case, $\mathcal{D}_2 = \mathcal{U}$. For any $D_1 \in \mathcal{D}_1$ and $D_2 \notin \mathcal{D}_1$, $|(\mathcal{D}_1 \setminus \{D_1\}) \cup \{D_2\}| = |\mathcal{D}_1| \geq k$. Therefore, the new set will also map to $\mathcal{U}$, of which D_1 is a member. In the second case, $\mathcal{D}_2 = \mathcal{D}_1$ and there will be no item D_2 satisfying the condition $D_2 \in \mathcal{D}_2$ and $D_2 \notin \mathcal{D}_1$. Symmetry is thus trivially achieved. □

A construction for (k, n) threshold data-based access control is defined by two operators:

- an encryption function Enc, taking as input n data-items $\mathcal{U} = \{D_1 ... D_n\}$, the value k and outputting a public string P;
- a decryption function Dec, which takes as input the public value P and k data-items $\mathcal{T} = \{T_1 ... T_k\}$, and yields $\mathcal{U}$ if and only if $\mathcal{T} \subseteq \mathcal{U}$.

(k, n) threshold data-based access control should satisfy the following security properties:

- knowledge of k items allows exact reconstruction of $\mathcal{U}$;
- knowledge of fewer than k items reveals no additional information (information theoretic security) or only a negligible amount of information (computational security) about any of the other individual items;
- applying Dec with a set of items that is partially correct should not give information about which items are correct.

We can distinguish between *direct* and *indirect* data-based access control. In the former, the remaining data-items can be recovered without any intermediate secret value that

protects the data. In the latter, knowledge of enough data-items allows for recovering a key for a conventional encryption algorithm that can be used to decrypt the full data-set. Because the latter relies on standard encryption techniques, the security will typically be computational rather than information theoretic.

The forms of data-based access control discussed here can be used in securing various types of databases. To enable this in practice, we need schemes that actually realise the desired properties. In the following, we will propose such schemes for (k, n) threshold data-based access control.

3 A Polynomial Interpolation Scheme

3.1 Shamir's Secret Sharing

Shamir [9] introduced a scheme for secret sharing based on polynomial curve fitting. Secret sharing involves dividing a secret D into n parts, of which k are necessary and sufficient to reveal the secret. The idea is to define a polynomial of degree $k-1$, using D as one of the coefficients and choosing the others randomly, and generate n points on the polynomial as the parts of the secret. Because the degree is $k-1$, k shares will suffice to reconstruct the coefficients and thus the secret.

In Shamir's scheme, the coefficients are chosen randomly, except for the coefficient that represents the secret. This makes it possible to limit the degree of the polynomial to k. However, in data-based access control, we have n data-items to fit into the polynomial. It is not possible to represent these as n points on a polynomial of degree $k-1$, because in general n points will not lie on a single polynomial of this degree.

3.2 Direct Scheme

We present two possible solutions. The first is to represent the data-items as n points uniquely defining a polynomial of degree $\leq n-1$. Assume that we have n data-items D_i $(1 \leq i \leq n)$ and an injective two-way function h mapping data-items to numbers below a prime p. We now define a polynomial f of degree $\leq n-1$ in $\mathbb{Z}_p$ by the n points $(i, h(D_i))$. If we then publish $n-k$ additional points on the polynomial, someone who knows k of the data-items can reconstruct the polynomial, the points and thereby the data-items.

We publish:

- n and p
- $n-k$ points $(j, f(j))$ on the polynomial, where $j > n$

Decryption can now be done by reconstructing the polynomial from the $n-k$ public points and k known points (n in total). One can then recalculate the remaining points, and by applying the inverse of h recover the data items. This requires that the data-items be encoded using a two-way function, which may only be possible in case of small data-items, such as passwords. If we wish to use a hash function to map the data-items to numbers, this approach is not feasible, because points cannot be converted back to data items. An indirect and more general approach, where an additional key is added, is defined below.

3.3 Indirect Scheme

Assume that we have n data-items D_i ($1 \leq i \leq n$) and a function h mapping data-items to numbers below a prime p. The function h may for example be a hash function, but this is not required for security. (The numbers must be uniformly distributed though, to prevent an adversary from acquiring information from correlations of data-items.) We choose a random key S from $\mathbb{Z}_p$. In general, $n + 1$ points are necessary and sufficient to define or reconstruct a polynomial of degree $\leq n$. We now define a polynomial f of degree $\leq n$ in $\mathbb{Z}_p$ by the $n + 1$ points $(0, S)$ and $(i, h(D_i))$.

Suppose that someone already knows $(i, h(D_i))$ for k of the data-items. In order to recover the full polynomial, including the key S, one needs $n + 1 - k$ additional points of the polynomial. Therefore, we publish:

- the data encrypted with key S
- n and p
- $n + 1 - k$ points $(j, f(j))$ on the polynomial, where $j > n$

The data may be accompanied by a plaintext description, allowing agents to find the data-set they want, as well as to specify the order of the items. Note that the value of k can be derived from n and the number of published points, and thus does not need to be published.

Since the $(n+1-k)$ points on the polynomial made public represent additional points on the polynomial, these will allow anyone who knows $(i, h(D_i))$ for k of the data-items to recover f from the $(n + 1 - k) + k = n + 1$ points in his possession. He can then recover the key S from the point $(0, S)$ and decrypt all the data. The additional points made public thus provide information about the relation between the data-items, without revealing additional information about the data-items themselves, as shown in [9].

Although it would be possible to use a polynomial of degree $n-1$ based on the points $(i, h(D_i))$ and then define $S \equiv f(0)$, this would mean that the key would depend on the data that the key is used to encrypt, leading to possible weaknesses in the encryption scheme employed in practice. We therefore choose to use a random key and polynomial of degree n.

3.4 Example

As an example, assume we have a data-set with the following description: Alice's passwords: (1, laptop), (2, mail), (3, social networking). We assume (without loss of generality) that the passwords are numbers. For explanation purposes, we use in the example the small values 24, 37 and 62 respectively. We also use $p = 67$ here. We choose as the secret key S the value 45. Now, we have the following points on the polynomial: $(0, 45)$, $(1, 24)$, $(2, 37)$, $(3, 62)$. Using a polynomial curve fitting algorithm, we can find that $f(x) = 41x^3 + 28x^2 + 44x + 45$. We wish to be able to recover one password as long as we remember the other two. Therefore, we publish two additional points on the polynomial: (4,10) and (5,60). This means that the following information is published:

- the description of the data-set: Alice's passwords: (1, laptop), (2, mail), (3, social networking)

- the data encrypted with key S (not shown here because of the simplicity of the example)
- $n = 3, p = 67$
- the additional points (4,10) and (5,60)

Now assume that Alice lost her password 3 (her social networking password). She remembers the other two passwords, though. She therefore possesses four points on the polynomial: $(1, 24)$, $(2, 37)$, $(4, 10)$ and $(5, 60)$. This is enough to obtain by polynomial curve fitting the polynomial $f(x) = 41x^3 + 28x^2 + 44x + 45$. Calculating $f(0) = 45$ gives Alice the key to decrypt all her passwords, including the missing one. Note that all computations can be done offline after obtaining the public data.

4 An Order-Invariant Scheme

For obtaining an order-invariant scheme, we apply the set difference construction of a secure sketch from [1]. For the direct scheme, assume again that we have an injective two-way function h mapping data-items to numbers below a prime p. Here, the secret values $h(D_i)$ are encoded as the x-coordinates for which the value of a polynomial f coincides with the value of a polynomial g. Since the values are encoded as x-coordinates as opposed to the y-coordinates in the previous scheme, their order is unimportant.

In [1], Reed-Solomon error correction is used to account for possibly incorrect set elements that are supplied in the decoding stage. Because we are not interested in error correction here, we can leave out this possibility. For decryption, one should input k values which are all correct. This allows us to simplify the scheme from [1] as follows.[1]

The encryption algorithm Enc now takes the following steps (again working in polynomials over $\mathbb{Z}_p$):

1. Choose a secret polynomial g of degree $k - 1$ at random;
2. Calculate and publish the unique monic polynomial f of degree exactly n by solving $f(h(D_i)) = g(h(D_i))$ for all D_i. Publish k as well.

Decryption now works as follows:

1. Reconstruct g by solving $f(h(D_i)) = g(h(D_i))$ for all known items;
2. Find the remaining values for which $f(h(x)) = g(h(x))$; these are the other data-items.

Since there is no error correction needed, there is minimal entropy loss. That is to say, given $k - 1$ values for which $f(x) = g(x)$, one learns nothing about the remaining root of $f(x) - g(x)$.

It is also possible to use this scheme for indirect data-based access control. In [1] it is shown that a secure sketch can be turned into a *fuzzy extractor*. A fuzzy extractor outputs a secure key based on the input of the secure sketch. When we turn the secure sketch, in our case (Enc, Dec), into a fuzzy extractor, we can thus obtain a key for indirect data-based access control. Alternatively, an extra data-item may be added that represents the key for decrypting the data (as we did in the ordered scheme).

[1] In a later version of their publication [2], the authors use a deterministic rather than a probabilistic variant of their algorithm. The following algorithms can be adapted accordingly.

5 Implementation

A prototype of the schemes discussed above has been implemented in Java, for demonstration purposes (approximately 2,000 LOC). The implementation includes polynomial interpolation as well as calculating the roots of polynomials in $\mathbb{Z}_p[x]$. For the latter, the Cantor-Zassenhaus probabilistic algorithm was used, which calculates the roots in $O(n^3 \log(p))$ operations in $\mathbb{Z}_p$ [10]. It is assumed that the polynomial is square-free, i.e. $D_i \neq D_j$. The implementation and source code will be made available upon request.

6 Extensions for Asymmetry

In the symmetric version of data-based access control, the encryption is symmetric: there is no distinction between keys and data in what one already knows. In this sense, any data may serve as part of a key for recovering other data. Also, it may be imposed that two or more keys be known in order to read certain data. Conversely, this also allows one to recover an additional key if one knows the other keys and the data. This is obviously not a universal solution; in some applications this symmetry is undesirable, for example when privacy-sensitive data is used to gain access. Also in traditional encryption, allowing access to data based on a key does not usually imply allowing access to the key based on the data. Privacy-sensitive data and keys may thus need additional protection, by introducing asymmetry in the policies.

There are at least two ways to introduce asymmetry into data-based access control:

- Give the data-items different weights. This can be done by multiple levels of indirect encryption, in which different combinations of data-items lead to different partial keys. The keys can then be combined to reconstruct the main key for decryption. For example, the policy could be that if you have a, b, c you can get d, e, and vice versa. Now we create three fuzzy extractors: one that creates a key from a, b, c, one that creates a key from d, e and one that creates the main key from any one of the aforementioned keys.
- Use the hashes to keep certain parts of the encryption one-way. If the function h mapping data-items to numbers is a hash function, we can assume that it is infeasible to reconstruct D_i from $h(D_i)$. If we do not include certain data-items in the published encrypted data, this directly implies that these data-items can serve only as credentials, and cannot be recovered by means of the secure sketch. Only the hash can be reconstructed. For example, the policy could be that if you have a and b, you can get c, if you have a and c, you can get b, but if you have b and c, you can't get a. If we use hashes and publish only the encryption of b and c, this policy is satisfied.

Note that the one-way property can also be achieved by multiple-level encryption. In the last example, one can again create three fuzzy extractors, in which the two upper-level ones create a key from a and b, and a and c, respectively. These ideas can be starting points for further research.

7 Strengths and Weaknesses

As far as we are aware, this paper is the first to introduce the idea of data-based access control. In a de-perimeterised world, this approach may help in putting the protection as close to the data as possible, i.e. having the data protect themselves. Compared to sticky policies and attribute-based encryption, this approach has the advantage that it does not need to rely on a separate key authority, because the data themselves allow for recovering the appropriate keys.

The fact that the security depends on data-items is both the main strength and the main weakness of this approach. In contrast to attribute-based encryption, we do not need to rely on a trusted key authority in order to distribute keys based on attributes, because in our case the attributes are data, and serve as keys themselves. This also means, however, that we cannot control the uniformity of the distribution of the key material. This means that the security is only as strong as the entropy in the data used. If $k = 2$ and it is easy to guess two of the data-items, the others are not secure anymore. It is therefore important to develop guidelines on which parameters to choose for the schemes (which items to combine in an encryption, the value of k).

In combination with fuzzy extractors [1], the approach may allow for retrieving data-sets of which items are only approximately known (each item may be slightly wrong). The fuzzy extractor is then used as the function h mapping data-items to numbers. For example, I may supply four passwords to recover my fifth one, and I am allowed make an error of 1 character in each of the passwords. Of course, this comes at the cost of some entropy loss. Also, Reed-Solomon error correction may be used to introduce the property that the *set* of required items is only approximately known (some items may be completely wrong). For example, I may supply four passwords, and if three of them are correct, I get all my five passwords.

After downloading the encrypted data-set, all computations can be done offline, without the need of a third party. Still, it may be possible that an adversary learns the value of a certain $h(D_i)$ without knowing D_i. For protection against this threat, one could use a salt in the function h, to make sure that $h(D_i)$ is different for each data-set the data-item is included in. Whether h could also be a probabilistic function may be a question for further research.

Moreover, the schemes introduced here may not be the most efficient. Research is therefore needed into alternative schemes and comparison of their properties in terms of security and efficiency. This research should also address the scalability of the different methods. In this paper, the aim was only to provide the framework and a proof-of-concept.

8 Conclusions

In this paper, we introduced the concept of data-based access control. This is a generalised form of de-perimeterised security in which the possibility for recovering protected data depends on the data one already knows. More specifically, we introduced the problem of recovering n data-items if one knows k of them, and not learning anything if one knows fewer than k. This problem is similar to a secure sketch, but it does

not inherently involve error correction, and the application area is different. We also introduced possible solutions to this problem based on polynomial interpolation and secure sketches, and suggested some applications. Further research in this new paradigm should reveal alternative and possibly more efficient schemes, as well as additional applications.

In the future, it would be useful to define other forms of data-based access control than (k, n) threshold schemes. Also, it should be investigated how applications, such as password recovery, can be implemented securely in practice. Another open question is how to deal with propagation of access: if an additional data item is recovered, this may in its turn give access to more data items. This may also affect revocation of policies: how can one be sure that users that should not have access anymore do not possess all the required data to gain access? Intuitively, one may add . Further research will investigate how this affects policies.

In this paper, we have focused on confidentiality as a security property. It may also be possible to use data-based security for integrity purposes, in the form of data-based signatures. An additional question that needs to be addressed is how signing with a certain key proves knowledge of the associated data.

Acknowledgements. This research is supported by the research program Sentinels (www.sentinels.nl). Sentinels is being financed by Technology Foundation STW, the Netherlands Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs. The authors wish to thank Pieter Hartel, André van Cleeff and Trajce Dimkov for useful comments on drafts of this paper.

References

1. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 523–540. Springer, Heidelberg (2004)
2. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM Journal on Computing 38(1), 97–139 (2008)
3. Ellison, C., Hall, C., Milbert, R., Schneier, B.: Protecting keys with personal entropy. Future Generation Computer Systems 16, 311–318 (2000)
4. Jericho Forum. Jericho whitepaper. Jericho Forum, The Open Group (2005)
5. Juels, A., Sudan, M.: A fuzzy vault scheme. Designs, Codes and Cryptography 38(2), 237–257 (2006)
6. Karjoth, G., Schunter, M., Waidner, M.: The platform for enterprise privacy practices: privacy-enabled management of customer data. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 69–84. Springer, Heidelberg (2003)
7. Mannan, M., van Oorschot, P.C.: Digital objects as passwords. In: 3rd USENIX workshop on hot topics in security (2008)
8. Sahai, A., Waters, B.: Fuzzy identity based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
9. Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)
10. Shoup, V.: A computational introduction to number theory and algebra, 2nd edn. Cambridge University Press, Cambridge (2008)