Self-Delegation with Controlled Propagation – or – What If You Lose Your Laptop

Oded Goldreich[*] Birgit Pfitzmann[†] Ronald L. Rivest[‡]

May 28, 1998

Abstract

We introduce delegation schemes wherein a user may delegate rights to himself, but may not safely delegate those rights to others. In our motivating application, a user has a primary (long-term) key that receives access rights that may not be delegated to others, yet the user may reasonably wish to delegate these rights to new secondary (short-term) keys he creates to use on his laptop when traveling, to avoid having to store his primary secret key on the vulnerable laptop. We propose several cryptographic schemes, both generic and practical ones, that fulfill these somewhat conflicting requirements, without relying on special-purpose (e.g., tamper-proof) hardware.

Keywords: Delegation, subkeys, key hierarchy, Zero-Knowledge Proofs, Knowledge Complexity, Threshold Schemes, gradual release of secrets, Signature Schemes, Non-Interactive Zero-Knowledge, Commitment Schemes.

[*]Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, ISRAEL. E-mail: oded@wisdom.weizmann.ac.il. Currently visiting LCS, MIT. Partially supported by DARPA grant DABT63-96-C-0018.

[†]Universität des Saarlandes, Fachbereich Informatik, D-66123 Saarbrücken, GERMANY. Email: pfitzmann@cs.uni-sb.de. Work partially done while visiting MIT and partially at the University of Hildesheim. Supported by DFG (German Research Foundation).

[‡]Laboratory for Computer Science, MIT, Cambridge, Mass., USA. Email: rivest@theory.lcs.mit.edu. Supported by DARPA grant DABT63-96-C-0018.

1 Introduction

In a networked world, users are represented by their public keys. A message whose digital signature verifies with a user's public key is assumed to have been signed by that user. Hence, users have to maintain the secrecy of their signing keys. In real life, however, they may wish to use relatively insecure computing devices, e.g., networked UNIX machines or laptops, for digital transactions. How can a user make use of such insecure devices while preserving the secrecy of his secret key?

One natural solution is for the user to use the secret key of his primary (long-term) key pair only on secure computing devices, and to create secondary (temporary, short-term) key pairs as needed for use on less secure devices. The user delegates a restricted amount of authority from the primary key pair to such a secondary key pair by signing a delegation certificate. This is simply a statement of the form "the secondary public-key $\langle \texttt{key2} \rangle$ can be used instead of the primary public-key $\langle \texttt{key1} \rangle$ for the following purposes and under the following conditions and/or limitations: $\langle \texttt{description} \rangle$". The statement is signed with the primary secret-key. This delegation certificate is transfered to the insecure device together with the secondary key pair, and is appended to all messages signed with the secondary key, so that the recipient may verify the delegation. If a secondary secret key is compromised, the loss is minimized because only a restricted amount of authority was delegated to it.

This is a good solution for many cases, but fails in cases where *personalized* rights are assigned to a *key*. The intention is to restrict the right to a specific person, and this is implemented by requiring a signature (or identification) relative to the key associated with this person. This person may indeed give away its (primary) secret-key, and thus the requirement "for personal use" is not really enforced. However, the assumption behind such schemes is that users are highly motivated not to give away their (primary) secret-key because this key allow others to do many more things in the user's name. Thus, the user is detered from propagating certain personalized rights to others by the fact that other rights will be automatically propagated along with the former rights. For example, a user may be willing to illegally propagate its personal subscription to some service, but not if this is linked to the propagation of the right to withdraw money from his[1] bank account. The *new* problem, introduced by using secondary keys in a setting where personalized rights are to be enforced, is that the users are no longer deterred from propagating personalized rights which they may delegate to secondary keys. This is the case since the delegation certificate may limit the rights of a secondary key to a specific right (e.g., subscription), thus leaving no additional risk for the user if he propagates this secondary key (and the subscription which goes with it). Hence, it seems that delegartion of rights to secondary keys and the enforcement of personalized rights are in conflict (see further discussion below). Resolving this conflict is the central goal of this paper. Put in other words, we ask: *How can delegation of rights be managed such that a user can delegate these rights to himself, but not to others?*

We consider two different approaches, which we term *statutory* and *transparent*.[2] Both approaches only approximate the above goal. Loosely speaking, in both approaches one aims at striking a balance between the following two conflicting requirements:

1. *Restricted damage from loss of secondary keys*: Loss of a *few* secondary secret-keys causes no (or little) damage to the user, beyond the loss and abuse by the finder or thief of the power delegated by the user to these secondary keys.

2. *Deterring propagation of secondary keys*: Giving away *many* secondary secret-keys to other users (e.g.,

[1] Throughout the paper we use the masculine form.

[2] In a sense these terms are a little misleading for several reasons. Firstly, any cryptographic scheme (including the "transparent" solutions presented in this work) obtains its sematics by a convention which should be viewed as statutory (e.g., a signature scheme is socially binding by a social or legal convention). Secondly, no cryptographic scheme is totally "transparent" and, in particular, the self-delegation schemes presented below do modify the underlying mechanism of secondary keys. Still, we believe these terms reflect the different nature of the approcahes presented below.

friends) greatly endangers the user in the sense that this may have consequences which are highly unpleasant to the user (and extend way beyond the rights associated with these secondary keys).

We comment that the slackness present in the above formulation seems inevitable since the finder/thief in the first requirement is "indistinguishable" from the friends in the second. Thus, the distinction is made quantitatively.

The paper focuses on the transparent approach (which may have applications beyond the scope of the self-delegation problem). The statutory approach is only discussed in the next subsection. We refrain from addressing the non-technical question of which of the approaches (statutory or transparent) is "better".

1.1 The Statutory Approach

Following the above paradigm, one may devise simple solutions in which trusted authorities (i.e., a judicial system) inflict suitable punishments on users which are proven to have propagated personalized rights to other users. The case against a user will consist of sufficiently many secondary-key; it is assumed that no user is going to incriminate himself, whereas other users may be less careful w.r.t not incriminating another user. Furthermore, users may even be encouraged (again by statutory means; e.g., awards) to incereminate other users. Such a judicial system is very flexible – it may consider special circumstances and determine the verdict based on these. On the other hand, such a system has a significant overhead.

Another type of statutory arrangement, which is closer in spirit to the solution presented next, is to define an action taken with sufficiently many secondary-keys (regardless of their assigned rights) as if it was taken by the user (i.e., as when the action is taken by the user's primary key). However, in such a case, one could still distinguish an action taken with sufficiently many secondary-keys from one taken with the actual primary key. Thus, this solution too is far from being "transparent".

1.2 The Transparent Approach

We propose to deter users from delegating personal rights to others by ensuring that the recipient must learn something about the primary secret key of the delegator. A user will then again be highly motivated *not* to delegate such rights so as not to risk giving up the fundamental rights associated with the primary secret key. The basic principle of all our proposals is that the user has to validate his secondary keys with respect to his primary secret key (in addition to signing a delegation certificate as above). Access requests signed with a secondary key will only be granted for secondary keys validated in this way.
It seems, however, that the following two goals are still in conflict:

1. Restricted damage from key disclosure: Minimize the damage caused by the inadvertent loss or theft of the user's current validated secret secondary key.
2. Controlled propagation: Discourage a user from delegating certain rights to others by ensuring that the recipient learns something about the user's primary secret key.

The conflict arises because the thief in the first goal is essentially indistinguishable from the recipient in the second goal. We strike the following balance between these requirements:

1. Loss of a *few* validated secondary secret keys will not endanger the security of the primary secret key. That is, they yield little or no knowledge about the primary secret key. Anyone who steals the secondary secret key of a user, even a few times, is therefore restricted to exactly the power the user delegated to these secondary keys.
2. *Many* validated secondary secret keys allow the recipient to recover the primary secret key efficiently. Thus, users are discouraged from distributing secondary keys to other parties on a large scale, because the others would gain power beyond what the user wanted to delegate to them with these secondary keys.

We present several constructions achieving such a balance. These constructions differ in the techniques used to encode the information about the delegator's primary secret key (in the secret validation information). Our constructions are categorized by the following parameters:

- Generic vs. specific: In Section 3, we present generic schemes which apply to any primary and secondary keys. They rely on non-interactive zero-knowledge proofs [4], and thus on the existence of trapdoor one-way permutations [17, 27]. In Section 4, we present more specific and efficient constructions where the primary key pair must belong to a discrete-logarithm based cryptographic scheme like Schnorr or DSS signatures [32, 15].
- Gradual vs. threshold: In Constructions 1 thru 1.2, the security of the primary secret key degrades gradually with the number of secondary secret keys available to an adversary until the primary secret key is totally revealed. In Constructions 2 thru 3.2, the primary secret key is fully secret up to a certain threshold number of secondary secret keys available to an adversary (e.g., 3, 10, or 100), but is easily recovered from any number of secondary secret keys larger than this or a second threshold.

  Additionally, some schemes allow the application to freely fix after how many secondary secret keys the adversary will get the primary secret key (i.e., the slope of the gradual degradation or the threshold, respectively), whereas in others this number is predefined by other system parameters, e.g., the key length.
- Limitations for secondary keys: We handle the limitations (or rights) associated with a secondary key in a general way. We call a description of these limits a limitations index. An archetypical example of limitation indices are time periods, such as days, during which a secondary key is valid. Other examples include upper bounds on value of transactions, specific services, etc.

  A relevant parameter for our schemes is the size of the domain of potential limitation indices. For example, if secondary keys are associated with days, which is indeed a very important case, then the domain of indices is relatively small. If, on the other hand, limitation indices stand for arbitrary specification of rights such as expressed by 1000-words English-text documents, then the domain is huge. The domains in Constructions 1.2 and 3.2 are unbounded, in Constructions 2 and 3.1 bounded, but exponentially large in the security parameter, and in Construction 1.1 relatively small. (Indeed, using appropriate hashing one may use schemes with an exponentially large domain of limitation indices as if their domain were unbounded.)

Note that we do not claim that all delegation should only be self-delegation – in many applications, delegation to others is explicitly desired; for example, to support organizational hierarchies, or at the discretion of the individual (e.g., the right to spend his own money or access rights to a certain *amount* of service). Our proposals are not in conflict with this: Both types of rights can be given on the same primary and secondary keys and even with the same delegation certificates. The only difference is that for those rights that should not be delegated to others, verifiers will require the presence of validation information related to the primary key of the *original* owner of the right, whereas in other cases they only require a delegation certificate.

1.3 Related Work

*** TO BE WRITTEN ***

2 The Model

Our model and constructions of self-delegation schemes only concern the core part of the applications described so far, and are therefore largely independent of the concrete application. In particular, we only deal with the primary and secondary keys, and not with certificates that third parties give to assign rights to the primary keys, such as the above-mentioned certificate asserting subscriberhood – these can be realized in standard ways, for example, by signatures with keys held by the third parties *on* the keys handled in the self-delegation scheme, and thus be added to our constructions in a modular way.

Furthermore, we do not explicitly consider the user's own authorization of the secondary keys, i.e., the delegation certificates (from primary key to secondary key) as described in the introduction. This is not quite as independent of our schemes as the statements by third parties, but whenever the underlying cryptographic scheme is a signature scheme, the delegation certificates can be made in a standard way with the primary secret key.

The notion of a generic scheme will more precisely mean the following: An *arbitrary* underlying cryptographic scheme may be given that contains an algorithm for generating primary key pairs and one for generating secondary key pairs, possibly depending on a primary key pair. The typical case is simply that an algorithm G for generating *one* key pair is given, and all primary and secondary key pairs are generated independently with this algorithm. Our mechanism allows to control propagation of such secondary keys without posing additional dangers on the underlying scheme and without making assumptions about how the keys of the underlying scheme may be used. For instance, signing (with a secret key) is not a zero-knowledge procedure, and in some protocols secret keys may even sometimes be revealed. Our controlled self-delegation schemes should not pose any *additional* risk to the keys, i.e., beyond the risk involved in using the given underlying cryptographic schemes. The schemes in Section 3 are of this type, and those in Section 4 are only slightly less generic in that they assume a specific generation algorithm for the primary secret key. We briefly mention variants that may be a little more efficient, but require that primary and secondary keys are only used in well-defined ways, e.g., within a specific signature scheme. In the rest of the model section, we concentrate on generic schemes.

We have to consider three types of parties: a server, which corresponds to a key certification infrastructure, users, who have primary and secondary keys, and verifiers, with whom the users interact using their secondary keys. A self-delegation scheme with controlled propagation has four protocols:

- Server setup: Here the server generates and publishes global parameters.
- Registration: Here the user, having generated a primary key pair $(\mathtt{sk}, \mathtt{pk})$ from an underlying cryptographic scheme, registers the primary public key with the server. We call the secret information that the user stores at the end of the registration his extended primary secret key, and the corresponding information published by the server the user's extended primary public key. For simplicity, we often omit the term "extended", and use $(\mathtt{sk}, \mathtt{pk})$ to denote also the extended key pair.
- Secondary key validation: Here the user, having generated a secondary key pair $(\mathtt{sk}_\ell, \mathtt{pk}_\ell)$ from the underlying cryptographic scheme, wants to validate it for a certain limitations index ℓ. He produces a validation tag, denoted $\mathtt{val}_\ell$, that he will use later to convince verifiers of the validity of the secondary key pair (when not having the primary secret key available).
- Verification: Here a user wants to convince a verifier, typically a service provider, that a certain public key $\mathtt{pk}_\ell$ can be used as a secondary key for the primary public key $\mathtt{pk}$.

  We stress that the user's secret inputs are only $\mathtt{sk}_\ell$ and $\mathtt{val}_\ell$; he no longer has the primary secret key available.

  The verifier also inputs the limitations index ℓ for which $\mathtt{pk}_\ell$ is supposed to be valid and global parameters published in server setup. Note that controlled propagation would make no sense if the verifier did not check ℓ; e.g., if ℓ denotes a specific date, the verifier must enter the current date rather than let the alleged user enter a date of his choice (which would always be the one for which the key is valid).

These protocols must be efficient, preferably as efficient as the underlying cryptographic scheme. Clearly, correctly validated secondary keys for correctly registered primary public keys should (almost) always be accepted by honest verifiers.

We call a triple $(\mathtt{sk}_\ell, \mathtt{val}_\ell, \mathtt{pk}_\ell)$ a correctly validated secondary key triple if it is a possible output of the correct secondary key generation and validation program. This is what a thief gets from an honest user. However, dishonest users who try to give away secondary keys to others are not restricted to correctly validated keys. We say that a user gives a second person a valid secondary key triple (for a public key $\mathtt{pk}$ and a limitations index ℓ) if the information enables the second person to convince an honest verifier almost certainly. Formally, this notion is only meaningful in connection with the second person's program, which may be arbitrary, yet must be efficient. For simplicity, we concentrate on users giving away full-quality secondary keys, but with definitions as in [1] this can be generalized to giving away information that will only convince the verifier with a certain probability.

Our two goals can now be formulated in more detail as follows:

- Restricted damage from key disclosure: If a certain user is honest, it should be infeasible for an adversary (thief) who only obtains *few* correctly validated secondary key triples to generate additional valid secondary key triples. Here, "additional" means for a limitations index ℓ^* for which the adversary has not obtained a correctly validated secondary key triple. The adversary should also not get too much knowledge about existing non-stolen secondary secret keys $\mathtt{sk}_\ell$ nor about the primary secret key. More precisely, as we concentrate on generic schemes, we have to speak about any knowledge gained by validation tags. A suitable terminology is that of knowledge complexity (cf., [26, 24]).[3]

  Specifically, gradual schemes have a parameter δ, called the slope, so that each correctly validated secondary key triple only gives δ bits of knowledge. Schemes with threshold have a parameter τ, called the threshold, so that less than τ correctly validated secondary key triples do not give the adversary any knowledge.

- Controlled propagation: If a user gives a second person *many* different valid secondary key triples, then the second person can efficiently compute the user's primary secret key. Here, "different" means for different limitations indices.

  In schemes with threshold, "many" typically means at least τ, but there are also schemes with a second threshold $\tau^* > \tau$, (i.e., allowing a gap). Gradual schemes have a similar parameter β, which will typically be about n/δ, where n is the key length.

For simplicity, we assume that all the security parameters δ, τ etc., are chosen at the server setup time. Typically, users are concerned with restricted damage from key disclosure and therefore wish τ to be large (or δ to be small), whereas service providers, mainly concerned with propagation control, want the opposite.

We consider the following types of attacks:

- The adversary for the first goal, "restricted damage", knows the user's primary public-key. It includes (both other users and) verifiers who have verified any number of this user's secondary public keys and with whom the user has then interacted using these keys, e.g., by signing messages.

  In a strong form of the requirement, the adversary also includes the server (i.e., the user may not want or need to trust the server).

[3] Loosely speaking, saying that a protocol (or a message) gives κ bits of knowledge with respect to a given state means that whatever can be efficiently computed after the execution of the protocol (or receipt of the message) can also be computed efficiently with probability at least $2^{-\kappa}$ from the state alone. Thus, for small κ, more precisely $\kappa = O(\log n)$ where n is the key length, if one can find the primary secret key with non-negligible probability after receiving at most κ bits of knowledge, then one can do the same without receiving these bits. (Recall that $f(n)$ is non-negligible if $f(n) \geq 1/\mathrm{poly}(n)$).

- The adversary for the second goal, "controlled propagation", consists of any number of colluding users and other verifiers. Typically, we only speak of "the user" because independence is clear. In some of our constructions achieving this goal depends on proper behaviour of the server (which is supposed to generate some random parameters).

In all the following schemes, the first goal, "restricted damage", is stated and proven with respect to a static adversary, i.e., the limitations indices of the revealed validation tags are determined beforehand rather than chosen adaptively by the adversary. We believe that all schemes are in fact adaptively secure, but refrain at this point from formulating and proving our belief. We also assume for the time being that each participant only carries out one protocol at a time (i.e., we do not consider protocol interleaving).

3 Generic Schemes

The basic idea is that each validation tag $\mathtt{val}_\ell$ contains some little "piece of information" σ_ℓ about the primary secret key $\mathtt{sk}$. The holder of a secondary key can only convince verifiers if he knows this σ_ℓ. Convincing the verifier must be done in zero-knowledge, so that verifiers do not gain an advantage in finding a user's primary secret key. On the other hand, knowledge of σ_ℓ's is related to the gradual release of a secret (as introduced for secret exchange in [3]): Knowledge of a few such pieces (i.e., σ_ℓ's) does not endanger the secret, whereas knowledge of many "orthogonal" pieces does endanger it. A crucial point is to ensure this "orthogonality" whenever many pieces are given to a second person in violation of the "controlled propagation" goal. To ensure this, the value of σ_ℓ is made to depend on the limitations index ℓ. Our constructions differ in the way this is achieved.

In the proof that he knows σ_ℓ, the user also has to prove that σ_ℓ is correct with respect to the primary secret key $\mathtt{sk}$, which is no longer available in the verification stage. Hence this part of the proof will be precomputed in the secondary key validation protocol. We use a non-interactive zero-knowledge proof system [4], more precisely one where one common random string Ref can be used for an arbitrary number of proofs [17]. (This result also applies to the more efficient basic construction in [27].) In schemes with a small set of limitations indices, we could improve efficiency by using a non-interactive zero-knowledge capable of only proving one assertion, and providing enough reference strings in advance.

Finally, we have to guarantee that all σ_ℓ's refer to the *same* primary secret key $\mathtt{sk}$. This is easy if the underlying cryptographic scheme guarantees that for every primary public key $\mathtt{pk}$, there is only one possible primary secret key $\mathtt{sk}$ and its validity can be verified in polynomial time. For example, this is the case with the prime factorization of RSA moduli or the discrete logarithm of an element in logarithm-based schemes. Yet, in other cases, additional information $\mathtt{aux}$ may be required to verify the validity of secrets with respect to publicly available information. Typically, the randomness used in the key-generation process suffices. We then denote by $\mathtt{pk}$ the entire information that is published. This motivates our interest in schemes where a 3-ary relation $\mathtt{Pred}$ with the following properties exists:

- NP-recognition of secret keys: $\mathtt{Pred}$ is recognizable in polynomial-time. This implies that the set of valid public keys
$$\mathtt{Pred}_1 = \{\mathtt{pk} | \exists \mathtt{sk}, \mathtt{aux} : (\mathtt{sk}, \mathtt{aux}, \mathtt{pk}) \in \mathtt{Pred}\}$$
and of valid key pairs
$$\mathtt{Pred}_2 = \{(\mathtt{sk}, \mathtt{pk}) | \exists \mathtt{aux} : (\mathtt{sk}, \mathtt{aux}, \mathtt{pk}) \in \mathtt{Pred}\}$$
are in NP and therefore have zero-knowledge proof systems [23].

- Unique secret keys: For every $\mathtt{pk}$, there exists at most one $\mathtt{sk}$ such that $(\mathtt{sk}, \mathtt{pk}) \in \mathtt{Pred}_2$.

- Key generation of the given cryptographic scheme produces triples $(\mathtt{sk}, \mathtt{aux}, \mathtt{pk}) \in \mathtt{Pred}$.

The following lemma shows that the requirement of unique secret keys is no serious restriction.

Lemma 1 *Suppose that one-way functions exist. Then, any cryptographic scheme where only computational secrecy of* sk *is required can be transformed into one with unique secret keys.*

Proof sketch: We use the commitment scheme from [28]. It is unconditionally binding, has a non-interactive opening stage, and can be based on any one-way function. In the commit stage, a random number is needed which may be chosen by a trusted server once and for all (see Section 5 in [28]). We modify the given cryptographic scheme by adding a commitment C on sk to the original public key pk and adding to aux the string $open$ needed to open the commitment. The other algorithms of the scheme are not changed.

Uniqueness is clear by the binding property of the commitment, and the additional commitment does not harm the computational secrecy of sk. □

3.1 Gradual schemes

The following construction is actually a framework for several slightly different versions with different properties.

Construction 1 (generic, gradual, framework): *Let a cryptographic scheme with unique secret keys with a relation* Pred *as above be given.*

- Server setup: *The server chooses the length n of the secret keys and the slope δ, where δ divides n. It randomly generates a reference string $Ref \in \{0,1\}^{\mathrm{poly}(n)}$ for the non-interactive zero-knowledge proof system, and sets up a scheme that defines a sequence $r_1, r_2, \ldots$ of n-bit strings so that anyone can obtain r_ℓ in $\mathrm{poly}(n, \log \ell)$ time. Several such schemes are discussed below.*
- Registration: *Suppose a user has generated a primary key pair $(\mathtt{sk}, \mathtt{pk})$, together with the value* aux *so that $(\mathtt{sk}, \mathtt{aux}, \mathtt{pk}) \in \mathtt{Pred}$. The user proves to the server in zero-knowledge that* pk *is valid, i.e., lies in* $\mathtt{Pred}_1$.
- Secondary key validation: *The validation tag for this user's secondary key w.r.t. the limitations index ℓ is the pair (σ_ℓ, π_ℓ) where*
  1. *σ_ℓ is the inner product in $\mathrm{GF}(2^\delta)$ of* sk *and r_ℓ, where both are considered as n/δ-dimensional vectors over $\mathrm{GF}(2^\delta)$.*
  2. *π_ℓ is a non-interactive zero-knowledge proof, with respect to the reference string Ref, that there exists* sk *so that $(\mathtt{sk}, \mathtt{pk}) \in \mathtt{Pred}_2$ and σ_ℓ is the inner product of* sk *and r_ℓ.*
- Verification: *The user gives the verifier an interactive zero-knowledge proof of knowledge [26, 1] that he knows (σ_ℓ, π_ℓ) such that π_ℓ is a valid non-interactive zero-knowledge proof as in Item (2) of the validation protocol. We stress that the common input to the interactive proof consists of Ref,* pk*, and r_ℓ.*

Proposition 1 (generic, gradual, framework): *Any instantiation of Construction 1, that is, no matter how the r_ℓ's are determined, has the following properties:*

1. Restricted damage from key disclosure: *The scheme leaks no knowledge except δ bits in each correct validation tag. The user only needs to trust the server for the randomness of the reference string Ref.*
2. Controlled propagation: *If the user gives a second person valid secondary key triples for a certain set of limitations indices, the second person can efficiently compute the inner product of one fixed primary secret key* sk *with the values r_ℓ for these limitations indices.*

Proof sketch: We first prove Part 1 ("restricted damage"). First, one easily sees that in the registration and verification protocols, the user does not give away any knowledge beyond the standard operation of the public-key infrastructure, because the rest of the protocols gives zero-knowledge proofs. Next, we consider the knowledge an adversary gains by obtaining a correctly validated secondary key triple. Recall that we only need to consider what is revealed on top of the secondary secret key itself, i.e., in the validation tag (σ_ℓ, π_ℓ). These are merely δ bits of knowledge, since σ_ℓ is only δ bits long and π_ℓ is a zero-knowledge proof referring to σ_ℓ and values the adversary knows already. (Recall that knowledge complexity is bounded by the actual length of objects and at most additive [24].)

We now prove Part 2 ("controlled propagation"). By the soundness of the interactive zero-knowledge proof of knowledge, having a valid secondary key triple (i.e., information that enables the second person to convince an honest verifier) implies that the second person knows a correct pair (σ_ℓ, π_ℓ). More formally, there is an effective way (called a knowledge extractor [1]) to reconstruct this pair given access to the second person's strategy and his auxiliary information. Now the soundness of the non-interactive zero-knowledge proof π_ℓ implies that σ_ℓ is in fact the inner product of the one fixed `sk` and r_ℓ. (Here the uniqueness property of $\mathtt{Pred}_2$ is used.) □

The scheme defining the sequence $r_1, r_2, \ldots$ should guarantee that sufficiently many of these values span almost the entire vector space, so that giving away many valid secondary keys compromises the primary key. Our first such scheme guarantees this in an optimal way, but can only be used for a domain of up to $2^\delta - 1$ limitations indices. This value is small because δ should be at most logarithmic in n to satisfy the restricted-damage condition. Thus, this scheme is most adequate for cases, like our motivating example where the limitations indices correspond to days; e.g., $\delta = 10$ allows 1023 days, after which the primary key pairs could be renewed.

Construction 1.1 (generic, gradual, severely bounded limitations indices): *We use Construction 1 with the following r_ℓ's: The server fixes an easily computable enumeration $\alpha_1, \ldots, \alpha_{2^\delta - 1}$ of the non-zero elements of $\mathrm{GF}(2^\delta)$. The value r_ℓ is the vector $(1, \alpha_\ell, \ldots, \alpha_\ell^{\beta-1})$, where $\beta = n/\delta$.*

Proposition 2 (generic, gradual, severely bounded limitations indices): *Construction 1.1 has the following controlled-propagation property: Any β valid secondary key triples yield the primary secret key.*

Proof sketch: Any β vectors r_ℓ form a Vandermonde matrix, which is always non-singular. Thus the system of equations that the second person has (by the general controlled-propagation property of Construction 1) is of full rank. The easily found solution is the primary secret key. □

To use Construction 1 with a larger domain of limitations indices, we essentially use uniformly random values r_ℓ. The server can set these up in different ways:

Construction 1.2 (generic, gradual, less bounded limitations indices): *We use Construction 1 with one of the following schemes for generating the r_ℓ's:*

1. For medium sized limitation set *The server generates and publishes, as part of the setup stage, a list of a certain number m of uniformly chosen $rand_\ell$.*

2. For unbounded limitation set: *The server provides a* "random oracle" (or Beacon) *service throughout the lifetime of the system; r_ℓ equals the answer of this random oracle on query ℓ. The random oracle may be implemented by the server by using a pseudorandom function [22].*

To analyze the controlled-propagation quality of these constructions, we need a technical lemma about the probability that too many of the r_ℓ's are linearly dependent. Note that when we talk of the rank of a matrix chosen from a subset of a field, we mean its rank over the field (i.e., the maximum number of linearly independent rows with respect to coefficients from the field).

Lemma 2 *Let m, n, t, R be integers so that $m > n > R$ and $m \geq t \geq R$, and q be a prime power. Let $S \subseteq \mathrm{GF}(q)$. Then, the probability $\mathsf{Pr}_S(m, n, t, R)$ that a uniformly chosen m-by-n matrix over S contains t rows which together have rank at most R is bounded above by*

$$\mathsf{Pr}_S(m, n, t, R) \ \leq \ \frac{(2m)^t}{t!} \cdot |S|^{-(n-R)\cdot(t-R)}.$$

The proof is given in the Appendix. For simplicity, we now restrict ourselves to the binary case, (i.e., $\delta = 1$).

Proposition 3 (generic, gradual, less bounded limitations indices): *For the first variant of Construction 1.2, with $\delta = 1$, the following controlled-propagation property holds: With probability at least $1 - 2^{-n}$, over the choice of the r_ℓ's, if a user gives away $2n$ valid secondary key triples to a second person, the second person can efficiently list a set of at most m^2 possibilities for the primary secret key. For the second variant of Construction 1.2, the above holds provided the user makes at most m oracle queries.*

The second person can use the above list in different ways: For example, he can efficiently perform any task requiring this secret key with probability $1/m^2$, by uniformly selecting an element of this list. Alternatively, in many applications, additional publicly available information can be used to determine which of these keys is correct.

Proof sketch: We wish to show that it is infeasible for the user to find $2n$ limitation indices so that the corresponding r_ℓ's, considered as vectors over $\mathrm{GF}(2)$, span a linear subspace of "small" dimension, more precisely smaller than $n - 2\log_2 m$. By the hypothesis, in both variants, the user can inspect at most m values r_ℓ's in this attempt. Invoking Lemma 2 with $q = 2, S = \{0, 1\}, t = 2n$, and $R = n - 2\log_2 m$, we obtain that the probability that the m inspected r_ℓ's, contain such a subset of $2n$ vectors is bounded by

$$\mathsf{Pr}_{\{0,1\}}(m, n, 2n, R) \ \leq \ \frac{(2m)^{2n}}{(2n)!} \cdot 2^{-(2\log_2 m)\cdot(n+2\log_2 m)} \ < \ 2^{-n}.$$

From this point on, we ignore this negligible probability event. Without loss of generality, we can assume that the user will only give away secondary key triples for limitations indices ℓ for which he has inspected r_ℓ (because inspecting $2n$ additional values is no significant overhead). Thus, we may assume that the corresponding values r_ℓ span a linear subspace of dimension at least $n - 2\log_2 m$. By the general controlled-propagation property of Construction 1, this implies that the second person has a system of at least $n - 2\log_2 m$ independent linear equations modulo 2 about the n unknown bits of `sk`. He can easily list all the (at most m^2) solutions of the system, which correspond to up to m^2 different possible values of `sk`. □

3.2 A treshold scheme

We now show a generic threshold construction. The threshold τ can be chosen very freely, i.e., between 1 and $2^n - 1$, where n is the length of the keys. The set of limitations indices is bounded, but the bound be much larger than in Construction 1.1: It is now 2^n.

Construction 2 (generic, threshold, exponentially large domain): *Let a cryptographic scheme with unique secret keys with a relation* `Pred` *as above be given. Additionally, we need a secure commitment scheme that is unconditionally binding.*

- *Server setup: The server chooses parameters n for the length of secret keys and τ for the threshold and a reference string $Ref \in \{0, 1\}^{\mathrm{poly}(n)}$ for the non-interactive zero-knowledge proof system. It fixes an easily computable enumeration $\alpha_1, ..., \alpha_{2^n-1}$ of the non-zero elements of $\mathrm{GF}(2^n)$.*

- *Registration: Suppose a user has generated a primary key pair* $(\mathtt{sk}, \mathtt{pk})$*, together with the value* aux *so that* $(\mathtt{sk}, \mathtt{aux}, \mathtt{pk}) \in \mathtt{Pred}$*. He first proves to the server in zero-knowledge that* pk *is valid, i.e., lies in* $\mathtt{Pred}_1$*. Next, he uniformly selects a polynomial of degree* $\tau - 1$ *over* $\mathrm{GF}(2^n)$ *with the primary secret key as the constant coefficient, say,* $p(x) \stackrel{\mathrm{def}}{=} \sum_{j=0}^{\tau-1} p_j \cdot x^j$ *with* $p_0 = \mathtt{sk}$*. This corresponds to setting up a secret sharing scheme as in [33]. He makes a commitment* C *to all the non-constant coefficients using the given commitment scheme. Typically, the length of the commitment automatically shows that the content is a polynomial of degree* $\tau - 1$*, otherwise this must be shown in zero-knowledge.*

  The user's extended primary public key is $(\mathtt{pk}, C)$*.*

- *Secondary key validation: The validation tag for this user's secondary key for the limitations index* ℓ *is the pair* (σ_ℓ, π_ℓ) *where*

  1. σ_ℓ *is the value of the polynomial* p *at the point* α_ℓ*;*
  2. π_ℓ *is a non-interactive zero-knowledge proof with respect to the reference string* Ref *that there exist* sk *and* p *so that* $(\mathtt{sk}, \mathtt{pk}) \in \mathtt{Pred}_2$*, the non-constant coefficients of* p *correspond to the commitment* C*, the constant term* p_0 *equals* sk*, and* $\sigma_\ell = p(\alpha_\ell)$*.*

- *Verification: The user gives a zero-knowledge proof that he knows a correct pair* (σ_ℓ, π_ℓ)*.*

Proposition 4 (generic, threshold, bounded limitations indices):

1. Restricted damage from key disclosure: *Public information and the validation tags of at most* $\tau - 1$ *correctly validated secondary key triples leak no knowledge. The user need not trust the server except for the randomness of the reference string* Ref*.*

2. Controlled propagation: *Any* τ *valid secondary key triples yield the primary secret key.*

Proof sketch: The second part follows as in Proposition 2. For Part 1, the security of the commitment scheme and the zero-knowledge proofs implies that the only non-negligible source of knowledge is in the values σ_i. These are $\tau - 1$ shares of a secret sharing scheme with threshold τ and therefore give no information about the secret pk [33]. □

4 Discrete-Logarithm Schemes

We now present two versions of a practical self-delegation scheme for cryptographic schemes based on discrete logarithms in groups of prime order. Important examples are Schnorr and DSS signatures [32, 15]. The advantage over the schemes in the previous section is that no general zero-knowledge proof techniques are needed, and the resulting self-delegation schemes are essentially as efficient as the underlying schemes. The secondary keys can even be from a different cryptographic scheme; only the primary key pair has to be from the discrete-logarithm-based scheme.

More precisely, we assume an underlying cryptographic scheme with the following key generation: First a prime q is chosen, typically 160 bits long; next a prime p, typically 512 bits, such that q divides $p - 1$; and then an element g of order q in the multiplicative group modulo p. A secret key is a uniformly chosen value $\mathtt{sk} \in \mathrm{GF}(q)^* = \{1, ..., q - 1\}$, and the corresponding main part of the public key is $h = g^{\mathtt{sk}} \bmod p$.

The basic idea in the following schemes is somewhat similar to the Verifiable Secret Sharing scheme based on discrete logarithms [18, 30]. Each valid secondary key triple will give one linear combination of the shares, such that enough secondary keys will yield an equation system that can be uniquely solved for the shares, and thus for the primary secret key. Both schemes have a threshold τ so that the validation tags of less than τ stolen secondary keys do not give any knowledge. Any $\tau \geq 2$ is possible.

Construction 3 (discrete log, threshold, framework): *Let any cryptographic scheme based on discrete logarithms in groups of prime order be given, i.e., the primary key generation is as described above.*

- *Server setup: For simplicity, we let the server generate the parameters (q, p, g) of the underlying scheme. It also sets up a scheme that defines a sequence $r_1, r_2, \ldots$ of τ-tuples over $\mathrm{GF}(q)$ so that anyone can obtain r_ℓ in $\mathrm{poly}(\tau, \log q, \log \ell)$ time. We write an individual tuple as $r_\ell = (r_{\ell,1}, \ldots, r_{\ell,\tau})$.*
- *Registration: Given a primary key pair $(\mathtt{sk}, \mathtt{pk})$ from the underlying discrete logarithm scheme, the user randomly selects a tuple $(s_1, \ldots, s_\tau)$ over $\mathrm{GF}(q)$ with the constraint $\mathtt{sk} = \sum_{j=1}^{\tau} s_j \bmod q$. This tuple is his extended primary secret key. His extended primary public key consists of the values $h_j = g^{s_j} \bmod p$, for $j = 1, \ldots, \tau$. The server verifies that $\mathtt{pk} = \prod_{j=1}^{\tau} h_j \bmod p$.*
- *Secondary key validation: The validation tag for this user's secondary key for the limitations index ℓ is the value $\mathtt{val}_\ell = \mathtt{sk}_\ell^*$ where*

  $$\mathtt{sk}_\ell^* = \sum_{j=1}^{\tau} r_{\ell,j} s_j \bmod q.$$

  Note that this validation tag can be seen as a secret key of the underlying discrete logarithm scheme; the corresponding public key would be $\mathtt{pk}_\ell^* = g^{\mathtt{sk}_\ell^*} \bmod p$.
- *Verification: The verifier can compute this value $\mathtt{pk}_\ell^*$ as*

  $$\mathtt{pk}_\ell^* = \prod_{j=1}^{\tau} h_j^{r_{\ell,j}} \bmod p$$

  using the values h_j in the user's extended primary public key. Now the user has to give a zero-knowledge proof of knowledge of $\mathtt{sk}_\ell^*$, i.e., of the discrete logarithm of $\mathtt{pk}_\ell^*$. The system from [7] is quite efficient, but it needs t rounds of communication (each with one exponentiation on both sides) for a security level of 2^{-t}. Actually, some of these can be executed in parallel with the usual trade-off between amount of parallelism and tightness of the security, cf. [20]. However, t need not be very large, because we do not try to exclude every single case of propagation anyway. An alternative is Schnorr identification [32], which needs only one exponentiation on each side, but is only known to be witness-hiding [19].

An alternative way of using this core scheme is to use the values $\mathtt{sk}_\ell^*$ directly as secondary secret keys, instead of as validation tags for independently generated secondary key pairs. This improves efficiency in cases where we can omit the proof of knowledge (because the subsequent usage of $\mathtt{sk}_\ell^*$ establishes knowledge of it). However, in such a case we have to be careful about how the security of the underlying scheme and our self-delegation scheme influence each other. In particular, the legitimate use of $\mathtt{sk}_\ell^*$ in the underlying scheme is typically not zero-knowledge and might therefore *additionally* endanger the primary secret key $\mathtt{sk}$ via our self-delegation scheme. Thus, security in such usage has to be considered specifically for each concrete underlying scheme (e.g., for Schnorr or DSS signatures). Specifically, for Schnorr identification scheme [32], security does hold (see below). However, the propositions below refer to the security of the core self-delegation scheme exactly as described in Construction 3.

We consider the following two instantiations of this framework:

Construction 3.1 (discrete log, threshold, exponentially large domain): *We use Construction 3 and define $r_{\ell,j}$ to be $\ell^{j-1} \bmod q$, while excluding $\ell = 1$* (because the equation for that case defines the primary secret key).

Construction 3.2 (discrete log, threshold, unbounded domain): *We use Construction 3 and let the server select a set $S \subset \mathrm{GF}(q)$ and chooses the values r_ℓ independently and uniformly as τ-tuples over S. Publication of these values is done in one of the ways described in Construction 1.2.*

However, the propositions below refer to the security The following two propositions describe the security of these instantiations. The first construction has stronger security: There is no gap between the thresholds in the two properties and no additional error probabilities. Moreover, similar to Constructions 1.1 and 2, there are none of the practical disadvantages of needing trusted random numbers. The advantages of the second construction are that it can be used for an unbounded set of limitations indices. Alternatively, this allows to use small $r_{\ell,j}$'s, which translates to improved efficiency as discussed below.

Proposition 5 (discrete log, threshold, exponentially large domain): *For Construction 3.1.*

1. Restricted damage from key disclosure: *The scheme leaks no knowledge to a thief who obtains less than τ correctly validated secondary key triples. The user need not trust the server.*
2. Controlled propagation: *Any τ valid secondary key triples yield the primary secret key.*

Proposition 6 (discrete log, threshold, unbounded domain): *For Construction 3.2.*

1. Restricted damage from key disclosure: *For any* given *set of at most $\tau' < \tau$ limitations indices, with probability $1 - |S|^{-(\tau-\tau')}$ (over the choice of the r_ℓ's), the scheme leaks no knowledge to a thief who obtains the correct validation tags for these limitations indices, The user has to trust the server for the randomness of the values r_ℓ.*
2. Controlled propagation: *Suppose the user can access at most a certain number $m < \sqrt{|S|}$ of the r_ℓ's. Then, with probability at least $1 - 2^{-\tau} \cdot |S|^{-1}$, if he gives a second person 2τ valid secondary key triples, the second person can easily derive the user's primary secret key.*

Proof sketches of these propositions are presented in Appendix B.

In Proposition 6, if the set of limitations indices is really unbounded and the bound on m is only given by the running-time of the user, we will simply use the set $S = \mathrm{GF}(q)$. Then the bound on m is realistic and the error probabilities in both properties are at most q^{-1}. If there is a fairly small set of limitations indices, denoted $\mathcal{L}$, we can choose $S = 0, ..., |\mathcal{L}|^2$ to have small exponents. Then, for reasonably small error probabilities, the threshold in the sense of the model will be what was called τ' above, i.e., τ will be chosen as $\tau' + \tau''$ such that $|\mathcal{L}|^{-2\tau''}$ and $2^{-\tau}|\mathcal{L}|^{-2}$ are considered small enough.

Efficiency: The efficiency of our discrete-logarithm self-delegation schemes can be summarized as follows: The use of a primary or secondary key pair, once it has been registered or verified, respectively, is exactly as in the underlying scheme. Registration (executed very rarely) requires τ exponentiations by the user, where τ is the threshold. Secondary key validation is essentially as efficient as in the underlying scheme. In verification of a secondary key, the basic part does not involve the user and needs τ exponentiations by the verifier. These exponentiations are to powers determined by the $r_{\ell,j}$'s, and thus it is desired to have these numbers relatively small. Here is where Construction 3.2 can become advantageous: The $r_{\ell,j}$'s in it reside in the set S, which may be set to $\{1, 2, ..., |S|\}$. In addition, verification (as described above) requires a zero-knowledge proof of knowledge of a discrete logarithm. However, in cases where the validation tag is used as a secondary key, the usage itself proves such knowledge and thus is for free. See example below.

Self-delegation in Schnorr's identification scheme: We now demonstrate the alternative usage of our self-delegation schemes, mentioned above. That is, we refer to the possibility of using the value $\mathtt{sk}_\ell^*$ itself as secondary secret key (rather than as a validation tag) associated with the limitation index ℓ, where $\mathtt{pk}_\ell^*$ is used as a corresponding public key. As an example we consider Schnorr's identification scheme [32], used in conjunction with the self-delegation of Construction 3.1.

Proposition 7 (Schnorr's identification, threshold, exponentially large domain): *Suppose that identification with respect to limitation index ℓ is conducted with respect to the public value $\mathtt{pk}^*_\ell$, determined in Construction 3.1. (The honest user wishes to identity itself uses $\mathtt{sk}^*_\ell$ as the secret key inside of Schnorr's identification protocol.)*

1. Restricted damage from key disclosure: *For a thief who obtains less than τ secondary secret keys of its choice, impersonation with respect to a different limitation index is as difficult as is impersonation in the original Schnorr scheme. We stress that this holds also when the thief can ask the user to repeatedly identify itself with respect to various limitation indices.*

2. Controlled propagation: *Any τ valid secondary secret keys yield the primary secret key.*

A proof sketch is presented in Appendix C.

References

[1] M. Bellare and O. Goldreich: On Defining Proofs of Knowledge; in *Crypto '92*, Springer-Verlag, LNCS Vol. 740, pp. 390-420, 1992.

[2] M. Bellare and P. Rogaway: Random oracles are practical: a paradigm for designing efficient protocols; in Proceedings of *1st Annual Conference on Computer and Communications Security*, ACM, pp. 62-73, 1993.

[3] M. Blum: How to Exchange (Secret) Keys; *ACM Transactions on Computer Systems*, Vol. 1, No. 2, pp. 175-193, 1983.

[4] M. Blum, P. Feldman, and S. Micali: Non-Interactive Zero-Knowledge and its Applications; in *20th STOC*, pp. 103-112, 1988.

[5] G. R. Blakley, C. Meadows, and G. B. Purdy: Fingerprinting Long Forgiving Messages; in *Crypto '85*, LNCS 218, Springer Verlag, Berlin 1986, pages 180-189.

[6] D. Boneh and J. Shaw: Collusion-Secure Fingerprinting for Digital Data; in *Crypto '95*, LNCS 963, Springer Verlag, Berlin 1995, pages 452-465.

[7] D. Chaum, J.-H. Evertse, and J. van de Graaf: An improved protocol for demonstrating possession of discrete logarithms and some generalizations; in *Eurocrypt '87*, Springer-Verlag, LNCS Vol. 304, pp. 127-141, 1988.

[8] D. Chaum, A. Fiat and M. Naor: Untraceable Electronic Cash; in *Crypto '88*, LNCS 403, Springer-Verlag, Berlin 1990, pages 319-327.

[9] D. Chaum: Showing credentials without identification: Transferring signatures between unconditionally unlinkable pseudonyms; in *Auscrypt '90*, LNCS 453, Springer-Verlag, Berlin 1990, pages 246-264.

[10] D. Chaum: Card-Computer Moderated Systems; *US Patent No. 4,926,480*, 15.05.1990.

[11] D. Chaum: Achieving Electronic Privacy; *Scientific American*, August, pp. 96-101, 1976.

[12] D. Chaum and T. P. Pedersen: Wallet Databases with Observers; in *Crypto '92*, LNCS 740, Springer Verlag, Berlin 1993, pages 89-105.

[13] W. Diffie and M. E. Hellman: New Directions in Cryptography; *IEEE Transactions on Information Theory*, Vol. 22, No. 6, pp. 644-654, 1976.

[14] C. Dwork, J. Lotspiech, and M. Naor: Digital Signets: Self-Enforcing Protection of Digital Information; in *28th STOC*, pp. 489-498, 1996.

[15] The Digital Signature Standard Proposed by NIST; *Communications of the ACM*, Vol. 35, No. 7, pp. 36-40, 1992.

[16] A. De Santis, M. Yung: Cryptographic Applications of the Non-Interactive Metaproof and Many-Prover Systems; in *Crypto '90*, Springer-Verlag, LNCS Vol. 537, pp. 366-377, 1993.

[17] U. Feige, D. Lapidot, and A. Shamir: Multiple non-interactive zero knowledge proofs based on a single random string; in *31st FOCS*, pp. 308-317, 1990.

[18] P. Feldman: A practical scheme for non-interactive verifiable secret sharing; in *20th FOCS*, pp. 427-437, 1987.

[19] U. Feige and A. Shamir: Witness Indistinguishability and Witness Hiding Protocols; in *22nd STOC*, pp. 416-426, 1990.

[20] A. Fiat and A. Shamir: How to Prove Yourself: Practical Solutions to Identification and Signature Problems; in *Crypto '86*, Springer-Verlag, LNCS Vol. 263, pp. 186-194, 1987.

[21] M. Gasser, A. Goldstein, Ch. Kaufman, and B. Lampson: The Digital Distributed System Security Architecture; in Proceedings of *12th National Computer Security Conference*, pp. 305-319, 1989.

[22] O. Goldreich, S. Goldwasser, S. Micali: How to Construct Random Functions; *Journal of the ACM*, Vol. 33, No. 4, pp. 792-807, 1986. Extended abstract in *25th FOCS*, 1984.

[23] O. Goldreich, S. Micali, and A. Wigderson: Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems; *Journal of the ACM*, Vol. 38, No. 1, pp. 691-729, 1991. Extended abstract in *27th FOCS*, 1986.

[24] O. Goldreich and E. Petrank: Quantifying Knowledge Complexity; in *32nd FOCS*, pp. 59-68, 1991. To appear in *Computational Complexity*.

[25] O. Goldreich, B. Pfitzmann and R.L. Rivest: Self-Delegation with Controlled Propagation – or – What If You Lose Your Laptop; Record 97-12 of the *Theory of Cryptography Library*, URL: `http://theory.lcs.mit.edu/~tcryptol`.

[26] S. Goldwasser, S. Micali, and C. Rackoff: The Knowledge Complexity of Interactive Proof Systems; *SIAM Journal on Computing*, Vol. 18, No. 1, pp. 186-208, 1989. Extended abstract in *17th STOC*, 1985.

[27] J. Kilian and E. Petrank: An Efficient Non-Interactive Zero-Knowledge Proof System for NP with General Assumptions; to appear in *Journal of Cryptography*. Available as TR95-038 of *ECCC (Electronic Colloquium on Computational Complexity)*, `http://www.eccc.uni-trier.de/eccc/`, 1995.

[28] M. Naor: Bit Commitment Using Pseudorandomness; *Journal of Cryptology*, Vol. 4, No. 2, pp. 151-158, 1991.

[29] T. Okamoto and K. Ohta: Divertible Zero-Knowledge Interactive Proofs and Commutative Random Self-Reducibility; in *Eurocrypt '89*, LNCS 434, Springer-Verlag, Berlin 1990, pages 134-149.

[30] T. P. Pedersen: Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing; in *Crypto '91*, Springer-Verlag, LNCS Vol. 576, pp. 129-140, 1992.

[31] B. Pfitzmann and M. Schunter: Asymmetric Fingerprinting; in *Eurocrypt '96*, LNCS 1070, Springer-Verlag, Berlin 1996, pages 84-95.

[32] C. P. Schnorr: Efficient Signature Generation by Smart Cards; *Journal of Cryptology*, Vol. 4, No. 3, pp. 161-174, 1991.

[33] A. Shamir: How to Share a Secret; *Communications of the ACM*, Vol. 22, No. 11, pp. 612-613, 1979.

[34] N. R.Wagner: Fingerprinting; in Proceedings of *IEEE Symposium on Security and Privacy*, pp. 18-22, 1983.

Appendix A: Proof of Lemma 1

The following lemma has to be proved:

Lemma 2 *Let m, n, t, R be integers so that $m > n > R$ and $m \geq t \geq R$, and q be a prime power. Let $S \subseteq \mathrm{GF}(q)$. Then, the probability $\mathsf{Pr}_S(m, n, t, R)$ that a uniformly chosen m-by-n matrix over S contains t rows which together have rank at most R is bounded above by*

$$\mathsf{Pr}_S(m, n, t, R) \leq \frac{(2m)^t}{t!} \cdot |S|^{-(n-R)\cdot(t-R)}.$$

Recall that arithmetic is always in a (prime) finite field, even though elements are selected from the subset S of the field.

Proof: Let $\mathsf{Pr}_S(n, t, r)$ denote the probability that a uniformly chosen t-by-n matrix over S has rank r. Then, our desired probability is upper bounded by

$$\mathsf{Pr}_S(m, n, t, R) \;\leq\; \binom{m}{t} \cdot \sum_{r=0}^{R} \mathsf{Pr}_S(n, t, r) \;<\; \frac{m^t}{t!} \cdot \sum_{r=0}^{R} \mathsf{Pr}_S(n, t, r). \tag{1}$$

Next we bound $\mathsf{Pr}_S(n, t, r)$ using the probability that $t - r$ rows (over S) are in the linear space spanned by r other rows. For any r-by-n matrix M, let Pr_S^M denote the probability that a uniformly selected n-dimensional vector over $S \subseteq \mathrm{GF}(q)$ resides in the linear subspace $\langle M \rangle$ spanned by M. We have

$$\begin{aligned}
\mathsf{Pr}_S(n, t, r) \;&\leq\; \binom{t}{r} \cdot \sum_{M \in S^{r \times n} : \mathrm{rk}(M) = r} \mathrm{Prob}(\text{matrix } M \text{ is chosen}) \cdot (\mathsf{Pr}_S^M)^{t-r} \\
&\leq\; \binom{t}{r} \cdot \max_{M \in S^{r \times n} : \mathrm{rk}(M) = r} \left\{ (\mathsf{Pr}_S^M)^{t-r} \right\}.
\end{aligned}$$

To bound Pr_S^M for a matrix M of rank r, we consider a set of $n - r$ row-reduced equations whose solution space is precisely $\langle M \rangle$. (Readers unfamiliar with dual vectorspaces may think of M as the generator matrix of a linear code and the equations as a corresponding checkmatrix. Row-reduced means that there is an identity submatrix, but not necessarily in adjacent columns.) The probability that a uniformly chosen vector over $S \subseteq \mathrm{GF}(q)$ fulfills these equations is at most $1/|S|^{n-r}$: First randomly set the r elements not corresponding to the identity submatrix. The remaining ones are then uniquely determined, and the solution may not even lie within S^n. We therefore get

$$\mathsf{Pr}_S(n,t,r) \;\leq\; \binom{t}{r} \cdot |S|^{-(n-r)\cdot(t-r)} \;\leq\; \binom{t}{r} \cdot |S|^{-(n-R)\cdot(t-R)}.$$

Substituting this into Eq. (1), we obtain

$$\mathsf{Pr}_S(m,n,t,R) \;\leq\; \frac{m^t}{t!} \cdot 2^t \cdot |S|^{-(n-R)\cdot(t-R)},$$

which proves the lemma. ■

Appendix B: Proofs of Propositions 5 and 6

We only present the more complicated proof sketch of Proposition 6. The proof of Propositions 5 can easily be derived using the fact that the system of equations defined by the r_ℓ's is a Vandermonde matrix.

For the proof of the first part, we first consider the probability that a specific vector, either r_{ℓ^*} or $(1,\ldots,1)$ (which plays a similar role by being the coefficients for sk), resides in the linear space spanned by the at most $\tau' < \tau$ vectors r_ℓ for the given limitations indices. This probability is easily upper bounded by $|S|^{-(\tau-\tau')}$ (as in the proof of Lemma 2). From now on, we assume that the other, more likely case occurs.

The core part of the proof is now that the entire view a thief gets is perfect zero-knowledge. More precisely, we consider the following protocol between a user and a thief (we have already omitted the zero-knowledge proof in verification):

- The common inputs are $\mathtt{pk}$ and the given r_ℓ's, and the user has the auxiliary input $\mathtt{sk}$.
- The user randomly shares $\mathtt{sk}$ as $(s_1,\ldots,s_\tau)$ as above.
- He publishes the extended primary public key and "sends" the k' validation tags for the given limitations indices.

We have to construct a simulator for this protocol. Let M be the matrix consisting of the τ' rows r_ℓ and one additional row $(1,\ldots,1)$. Without loss of generality, we can now assume that $\tau' = \tau - 1$ and M is of rank τ (otherwise we can give the thief additional information to bring him into this situation). Anyone can easily compute the inverse M^{-1}. For each given vector of the $\tau - 1$ tags and the secret key, one can easily compute the only possible corresponding tuple $(s_1,\ldots,s_\tau)$ by multiplying the given vector with M^{-1}. The simulator works as follows:

1. It randomly selects tags $\mathtt{sk}^*_\ell$ for the $\tau - 1$ given limitations indices. For simplicity, we denote these limitations indices by $1,\ldots,\tau-1$.
2. It computes $\mathtt{pk}^*_\ell = g^{\mathtt{sk}^*_\ell}$ for $\ell = 1,\ldots,\tau-1$.
3. Not knowing $\mathtt{sk}$, it applies M^{-1} to the exponentiated values instead of the discrete logarithms. This yields equations of the following form

   $$h_j \;=\; g^{s_j} \;=\; g^{(\sum_{\ell<\tau} m^*_{j,\ell}\mathtt{sk}^*_\ell)+m^*_{j,\tau}\mathtt{sk}} \;=\; \left(\prod_{\ell<\tau}(\mathtt{pk}^*_\ell)^{m^*_{j,\ell}}\right)\cdot \mathtt{pk}^{m^*_{j,\tau}},$$

   where the $m^*_{j,\ell}$'s are the elements of M^{-1}.
4. It publishes these values h_j and the originally chosen tags.

It is easy to see that the results of this simulator are distributed exactly like those of the original protocol. This ensures that the self-delegation scheme does not endanger the security of the underlying scheme.

To prove the second part, we first invoke Lemma 2 with the given set S, $n = \tau$, $t = 2\tau$ and $R = \tau - 1$: The probability that the m inspected r_ℓ's contain a subset of 2τ vectors which span a linear subspace of dimension at most $\tau - 1$ is bounded by

$$\mathsf{Pr}_S(m, \tau, 2\tau, \tau - 1) \;\leq\; \frac{(2m)^{2\tau}}{(2\tau)!} \cdot |S|^{-(\tau+1)} \;\leq\; \frac{2^{2\tau}}{(2\tau)!} \cdot |S|^{-1} \;<\; 2^{-\tau} \cdot |S|^{-1}.$$

Otherwise, we may assume that the r_ℓ's used in the validation of the 2τ secondary keys span a linear subspace of dimension τ. The second person therefore has a system of τ independent linear equations over $\mathrm{GF}(q)$ in the τ unknown values s_j, which he can easily solve.

Appendix C: Proof of Proposition 7

Let us first recall *Schnorr's identification* protocol [32], which is actually a proof of knowledge of a discrete logarithm.

Parameters: primes p, q and an element, g, of order q in $\mathbb{Z}_p^*$. All known to both parties.

Secret key: $s \in \mathbb{Z}_q$, held by the prover.

Public key: $h = g^s \bmod p$, held by the verifier.

Protocol for proving knowledge of s.

1. The prover uniformly selects $x \in \mathbb{Z}_q$, computes $y = g^x \bmod p$ and sends y to the verifier.
2. The verifier uniformly selects (a challenge) $\alpha \in \mathbb{Z}_q$, and sends it to the prover.
3. The prover responds with $\beta = \alpha \cdot s + x \bmod q$.
4. The verifier accepts if and only if

$$g^\beta \equiv h^\alpha \cdot y \pmod{p}$$

Let us next provide a full description of the *self-delegation identification scheme* referred to in Proposition 7.

- Server setup: The server generates the parameters (q, p, g) referred to above. It also adopts the convention by which $r_\ell = (r_{\ell,1}, ..., r_{\ell,\tau})$ satisfies $r_{\ell,j} = \ell^{j-1} \bmod q$.
- Registration: Exactly as in Construction 3. The user's (extended) primary secret key is a tuple $(s_1, ..., s_\tau)$ over $\mathbb{Z}_q$, and the corresponding (extended) primary public key consists of the tuple $(h_1, ..., h_\tau)$, where $h_j = g^{s_j} \bmod p$, for $j = 1, ..., \tau$.
- Generation of secondary keys: The secondary secret-key associated with the limitations index ℓ is the value

$$\mathtt{sk}_\ell = \sum_{j=1}^{\tau} r_{\ell,j} s_j \bmod q.$$

  The corresponding secondary public-key is

$$\mathtt{pk}_\ell = \prod_{j=1}^{\tau} h_j^{r_{\ell,j}} \bmod p$$

  and indeed $\mathtt{pk}_\ell \equiv g^{\mathtt{sk}_\ell} \pmod{p}$.

- ID–Verification (w.r.t limitation ℓ): The verifier can compute the value $\mathtt{pk}_\ell$ as above (using the values h_j's in the user's extended primary public key). The parties invoke Schnorr's protocol, where the prover enters with (i.e., sets the local input s to equal) the secret-key $\mathtt{sk}_\ell$, and the verifier enters with (i.e., sets the local input h to equal) the secret-key $\mathtt{pk}_\ell$.

We show that any restricted (as in the proposition) attack, A, against the above self-delegation identification can be transformed into an as successful attack, A', against Schnorr's original scheme.[4] Consider such an attack, A. It is given upto $\tau - 1$ secondary secret-keys, corresponding to some limitations $\ell_1, ..., \ell_{\tau-1}$. In addition, the attacker can ask the user to identify itself with respect to any limitation index, for several times. At the end, the attacker tries to impersonate the user with respect to some limitation index $\ell^* \notin \{\ell_1, ..., \ell_{\tau-1}\}$. For sake of simplicity, we assume that $\ell_j = j$, for $j = 1, ..., \tau - 1$. We now describe an attack A' against Schnorr's original scheme.

The attack A'. The input is the user's public-key $h \in \mathbb{Z}_p^*$ (where the user's secret-key, s, satisfies $g^s \equiv h \pmod p$). The attack starts by setting the grounds to an emulation of the attack A. Towards this end, A' uniformly selects $\mathtt{sk}_1, ..., \mathtt{sk}_{\tau-1} \in \mathbb{Z}_p^*$ (so that it can hand these as secondary secret-keys to A). Next, it sets $\mathtt{pk}_j = g^{\mathtt{sk}_j} \bmod p$, for $j = 1, ..., \tau - 1$, and $\mathtt{pk}_\tau = h$. (The latter effectively sets $\mathtt{sk}_\tau = s$.)

The next step is determining the inverse linear transformation from the secondary keys to the primary key. Recall that we have

$$\mathtt{sk}_\ell = \sum_{i=1}^{\tau} r_{\ell,i} s_i \quad \text{for } \ell = 1, .., \tau$$

and so, inverting the matrix of $r_{\ell,i}$'s we obtain a matrix of $c_{i,\ell}$'s so that

$$s_i = \sum_{\ell=1}^{\tau} c_{i,\ell} \mathtt{sk}_\ell \quad \text{for } i = 1, .., \tau$$

Using the same $c_{i,\ell}$'s we have

$$h_i = \prod_{\ell=1}^{\tau} \mathtt{pk}_\ell^{c_{i,\ell}} \quad \text{for } i = 1, .., \tau$$

and so, A', sets the h_i's like that.

Now, A' invokes A providing it with the (extended) public-file $(h_1, ..., h_\tau)$ and with the $\tau - 1$ "disclosed" secondary secret-keys, $\mathtt{sk}_1, ..., \mathtt{sk}_{\tau-1}$. Observe that A is invoked on distribution identical to the one for which it is defined (as an adversary strategy). The strategy A now makes requests to participate as verifier in self-delegation identification with respect to limitations of its choice. These requests are handled by letting A' make requests for identification in Schnorr's scheme. Specifically, suppose that A requests an identification with respect to ℓ (which, w.l.o.g., is not in $\{1, ..., \tau - 1\}$)[5]. Then A' invokes Schnorr's scheme playing the role of verifier. It thus obtains from the user some $y \in \mathbb{Z}_p$ which it forwards to A. When A responds with the challenge α, algorithm A' sets its challenge to

$$\alpha' = \sum_{i=1}^{\tau} r_{\ell,i} c_{i,\tau} \cdot \alpha \bmod q$$

When the user responds with $\beta' = \alpha' \cdot s + x \bmod q$, where $y \equiv g^x \pmod p$, algorithm A' answers A with

$$\beta = \beta' + \sum_{j=1}^{\tau-1} \sum_{i=1}^{\tau} r_{\ell,i} c_{i,j} \mathtt{sk}_j \cdot \alpha \bmod q$$

[4] Our argument resembles diverted and blinded proofs which have first appeared in [29, 12].

[5] There is no point in asking for verification w.r.t limitations for which the adversary knows the secret-keys. Indeed, such requests can be granted by direct emulation using the corresponding secret-key.

We comment that in case $\ell = \tau$, both $\alpha' = \alpha$ and $\beta = \beta'$. Recalling that $s = \mathtt{sk}_\tau$, we show that indeed $\beta \equiv \alpha \cdot \mathtt{sk}_\ell + x \pmod q$, and so that the response to A is correct. Specifically,

$$\begin{aligned}
\beta &\equiv \beta' + \sum_{j=1}^{\tau-1} \sum_{i=1}^{\tau} r_{\ell,i} c_{i,j} \mathtt{sk}_j \cdot \alpha \pmod q \\
&\equiv \sum_{i=1}^{\tau} r_{\ell,i} c_{i,\tau} \cdot \alpha \cdot \mathtt{sk}_\tau + x + \sum_{j=1}^{\tau-1} \sum_{i=1}^{\tau} r_{\ell,i} c_{i,j} \mathtt{sk}_j \cdot \alpha \pmod q \\
&\equiv x + \alpha \cdot \sum_{i=1}^{\tau} \sum_{j=1}^{\tau} r_{\ell,i} c_{i,j} \mathtt{sk}_j \pmod q \\
&\equiv x + \alpha \cdot \sum_{i=1}^{\tau} r_{\ell,i} s_i \pmod q \\
&\equiv x + \alpha \cdot \mathtt{sk}_\ell \pmod q
\end{aligned}$$

When A announces that it wants to try to impersonate with respect to some limitation ℓ^*, algorithm A' announces an attempt to impersonate in the basic Schnorr scheme. Algorithm A' obtains from A its first message, some $y \in \mathbb{Z}_p$, which A' forwards to the verifier it wishes to fool. When the verifier responds with a challenge α', algorithm A' forwards $\alpha = \alpha' / \sum_{i=1}^{\tau} r_{\ell^*,i} c_{i,\tau} \bmod q$ to A, and when the latter responds with a β, algorithm A' forwards $\beta' = \beta - \sum_{j=1}^{\tau-1} \sum_{i=1}^{\tau} r_{\ell^*,i} c_{i,j} \mathtt{sk}_j \cdot \alpha \bmod q$ to the verifier. Note that in case A succeeds in its impersonation attack, $g^\beta \equiv \mathtt{pk}_{\ell^*}^\alpha y \pmod p$ holds, and using a manipulation as above it follows that $g^{\beta'} \equiv h^{\alpha'} y \pmod p$ holds too. Specifically,

$$\begin{aligned}
g^{\beta'} &\equiv g^\beta \cdot g^{-\sum_{i=1}^{\tau} \sum_{j=1}^{\tau-1} r_{\ell^*,i} c_{i,j} \mathtt{sk}_j \cdot \alpha} \pmod p \\
&\equiv \mathtt{pk}_{\ell^*}^\alpha y \cdot g^{-\sum_{i=1}^{\tau} r_{\ell^*,i} s_i \cdot \alpha} g^{\sum_{i=1}^{\tau} r_{\ell^*,i} c_{i,\tau} \mathtt{sk}_\tau \cdot \alpha} \pmod p \\
&\equiv y \cdot \mathtt{pk}_{\ell^*}^\alpha \cdot g^{-\mathtt{sk}_{\ell^*} \cdot \alpha} \cdot h^{\sum_{i=1}^{\tau} r_{\ell^*,i} c_{i,\tau} \cdot \alpha} \pmod p \\
&\equiv y \cdot h^{\alpha'} \pmod p
\end{aligned}$$

We conclude that whenever A succeeds in its impersonation attack against the self-delegation identification scheme, A' succeeds in its attack on the basic Schnorr scheme.