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

Oded Goldreich* Birgit Pfitzmann† Ronald L. Rivest‡

August 7, 1998

Abstract

We introduce delegation schemes wherein a user may delegate certain rights to himself, but may not safely delegate these rights to others. In our motivating application, a user has a primary (long-term) key that receives some personalized access rights, 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 ones under general assumptions and more specific 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 $\langle$`key2`$\rangle$ can be used instead of the primary public key $\langle$`key1`$\rangle$ for the following purposes and under the following conditions and/or limitations: $\langle$`description`$\rangle$". The statement is signed with the primary secret key. This delegation certificate is transferred 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*. Here 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 allows others to do many more things in the user's name. Thus, the user is deterred 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 delegation 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?*

Corresponding to our motivating example – the use of insecure computing devices – we only consider solutions that do not employ special-purpose hardware (e.g., tamper-resistant or with biometrics). We consider two different approaches, which we term *statutory* and *transparent*. The latter means that the measures are invisible to users except for the unavoidable fact that issuers, holders, and verifiers of a personalized right must be aware that this right is personalized.

Both approaches only approximate the above goal. Loosely speaking, they do this by fulfilling the following two weaker requirements (with respect to secondary keys to which personalized rights are delegated):

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 possible abuse (by the finder or thief) of the power delegated by the user to these secondary keys.

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

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

The slackness present in this 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, where the solutions are cryptographic. (These cryptographic schemes may have applications beyond the scope of the self-delegation problem.) The statutory approach, which has non-cryptographic solutions, is only discussed in the next subsection. We refrain from addressing the non-technical question of which of the approaches is "better". We note, however, that the solutions in the two approaches differ in several aspects, and all aspects should be taken into account when evaluating these solutions.

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 others. The case against a user will consist of *sufficiently many* secondary secret keys (with corresponding certificates assigning personalized rights); it is assumed that no user is going to incriminate himself, whereas other users may be less careful not to incriminate another user. Furthermore, users may even be encouraged (again by statutory means; e.g., awards) to incriminate others. 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, closer in spirit to the solution presented next, is to define an action taken with *sufficiently many* validated secondary secret key (regardless of their assigned rights) as if it was taken by the user (i.e., with the corresponding primary key). However, in such a case, all verifiers of any signatures, even those that have nothing to do with personalized rights, must be aware of this arrangement. Thus, this solution too is not transparent.

1.2 The Transparent Approach

In the transparent approach, users are deterred from delegating personalized rights to others by ensuring that the recipient learns 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 an ordinary delegation certificate). As described below, the validation process will deter the user from delegating personalized rights to others. Use of personalized rights is granted only when one proves possession of a *validated* secondary-key (or possession of the primary-key, as usual), mere possession of a secondary-key (without corresponding validation) will not do.

The conflicting goals stated above, are thus rephrased as follows:

1. Restricted damage from loss of secondary keys: 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.

2. Deterring propagation of secondary keys: *Many* validated secondary secret keys allow the recipient to recover the primary secret key efficiently.

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 can therefore be built from any trapdoor one-way permutations [16, 24]. 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 [30, 15].
- Gradual vs. threshold: In Constructions 1 thru 3, 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 4 and 5, 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: Apart from honest users, service providers will wish to limit the rights associated with secondary keys to ensure that many different keys are needed for large-scale illegal propagation of rights. We handle these limitations associated with a secondary key in a general way, and call their description a limitations index. An archetypal example of limitations indices are time periods, such as days, during which a secondary key is valid. Other examples include upper bounds on value of transactions, lists of specific services, etc.

  A relevant parameter for our schemes is the size of the domain of potential limitations indices. For example, if secondary keys are associated with days, the domain of indices is relatively small. If, on the other hand, limitations indices stand for arbitrary specification of rights such as expressed by 1000-words English-text documents, then the domain is huge. The domain in Construction 3 is unbounded, in Constructions 4 and 5 bounded, but exponentially large in the security parameter, and in Construction 2 relatively small. (Indeed, using appropriate hashing, one may use schemes with an exponentially large domain of limitations indices as if their domain were unbounded.)

Our proposals for self-delegation of personalized rights are not in conflict with normal delegation of other types of rights: 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 personalized rights, 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

Delegation certificates are merely documents which assert that certain rights associated to a specific public key are to be passed to another public key. Such documents are verifiable with respect to the former public key, and are binding (as all digital signatures) under certain statutory provisions. Technical problems with such certificates occur where one makes additional requirements (e.g., anonymity, standardization of contexts, or non-transferability) regarding such certificates. None of these extensions is required in our case.

In our case the goal is to *restrict* the delegation of some rights; specifically, of personalized rights. Thus, ordinary delegation certificates will not do, unless they are augmented by either an auxiliary statutory arrangement or auxiliary validation information, as discussed above. (In fact, one may view the augmentation of delegation certificates, suggested above, as a new type of certificate which one may call a self-delegation certificate.) Other approaches to the issue of limiting the delegation of rights were suggested in [7, 8].

We note the analogy between the current work and [14]: The latter paper presents a transparent approach to copyright protection, in contrast to the (typical) statutory approach based on fingerprinting [32, 5, 6, 29].

Some analogy exists also between self-delegation schemes and $(\tau - 1)$-spendable coins [10, 27]. Conceptually speaking, the problems are different since there is no requirement on personal use in the latter, and the main issue is untraceability (otherwise identification of overspenders would be trivial). Technically speaking, one can see that the analogy does not extend too far since here we construct self-delegation schemes based on standard complexity assumptions, whereas such schemes are not known for untraceable cash.

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 $\mathtt{pk}$ in the self-delegation scheme.

  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. [23, 22]).[2]

  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,

[2]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 $f(n) \geq 1/\mathrm{poly}(n)$).

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 in server setup. 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).

- 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.[3] 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

[3] We retract an informal statement made in previous versions that we believe they are in fact also adaptively secure.

of proofs [16]. (This result also applies to the more efficient basic construction in [24].) In schemes with a small set of limitations indices, we could improve efficiency by using a non-interactive zero-knowledge proof system 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 [21].
- 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 $(\mathtt{sk}, \mathtt{aux}, \mathtt{pk}) \in \mathtt{Pred}$.
- Validity: If $(\mathtt{sk}, \mathtt{pk}) \in \mathtt{Pred}_2$, then there exists $\mathtt{aux}$ such that $(\mathtt{sk}, \mathtt{aux}, \mathtt{pk})$ is in the range of the key-generation algorithm.

For specific schemes, a weaker validity condition may suffice. For instance, for signature schemes it is sufficient if $(\mathtt{sk}, \mathtt{pk}) \in \mathtt{Pred}_2$ implies that signing with $\mathtt{sk}$ yields valid signatures with respect to $\mathtt{pk}$. This may sometimes lead to more efficient membership proofs, while maintaining the same level of motivation for the user not to give such an $\mathtt{sk}$ away. 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* $\mathtt{sk}$ *is required can be transformed into one with unique secret keys. The keys in the transformed scheme are augmentations of keys generated with the original key generation, and the other algorithms of the scheme are not changed, i.e., they ignore the new components of the keys.*

Proof sketch: We use the commitment scheme from [25]. 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 R is needed which may be chosen by a trusted server once and for all (see Section 5 in [25]). We modify the given cryptographic scheme by adding a commitment C on $\mathtt{sk}$ to the original public key $\mathtt{pk}$ and defining $\mathtt{aux}$ to be the pair (r_1, r_2) of random strings that were used in the generation of $(\mathtt{sk}, \mathtt{pk})$ and in generating the commitment from $\mathtt{sk}$ and R, respectively. The predicate $\mathtt{Pred}$ is defined to contain all triples $(\mathtt{sk}, (r_1, r_2), (\mathtt{pk}, C))$ where $\mathtt{sk}, \mathtt{pk}$, and C are generated correctly using r_1 and r_2.

Uniqueness is clear by the unconditional binding property of the commitment, and the additional commitment does not harm the computational secrecy of $\mathtt{sk}$. Validity follows immediately from the definition of $\mathtt{Pred}$. $\square$

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* $\mathtt{Pred}$ *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 $r_1, r_2, ...$ 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* $\mathtt{aux}$ *so that* $(\mathtt{sk}, \mathtt{aux}, \mathtt{pk}) \in \mathtt{Pred}$*. The user proves to the server in zero-knowledge that* $\mathtt{pk}$ *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* $\mathtt{sk}$ *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* $\mathtt{sk}$ *so that* $(\mathtt{sk}, \mathtt{pk}) \in \mathtt{Pred}_2$ *and σ_ℓ is the inner product of* $\mathtt{sk}$ *and r_ℓ.*

- Verification: *The user gives the verifier an interactive zero-knowledge proof of knowledge [23, 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. The common input to the interactive proof consists of Ref,* $\mathtt{pk}$*, 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, L, then the second person can efficiently compute the inner product of one fixed primary secret key* $\mathtt{sk}$ *with each value r_ℓ corresponding to $\ell \in L$.*

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 [22].)

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.) Finally, the validity property guarantees that this `sk` is a correct primary secret key. □

The scheme defining the sequence $r_1, r_2, ...$ 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 2 (generic, gradual, severely bounded limitations indices): *We use Construction 1 with the following r_ℓ's: The server fixes an easily computable enumeration $\alpha_1, ..., \alpha_{2^\delta - 1}$ of the non-zero elements of* $\mathrm{GF}(2^\delta)$. *The value r_ℓ is the vector $(1, \alpha_\ell, ..., \alpha_\ell^{\beta-1})$, where $\beta = n/\delta$.*

Proposition 2 (generic, gradual, severely bounded limitations indices): *Construction 2 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 linear 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 3 (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 limitations set: *The server generates and publishes, as part of the setup stage, a list of a certain number m of uniformly chosen values r_ℓ.*
2. For unbounded limitations 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 [20].*

In practice, one may be tempted to use a concrete publicly available function, typically a hash function, believed to "behave randomly" (cf. [19, 2]). However, in connection with the role that m, the bound on the number of function values an adversary can evaluate, plays in the proposition below, it would need to be a very slow function. 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 3, 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 3, 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 $2n$ limitations 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. 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 and we could then work with $m' = m + 2n$). 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. 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 $\mathtt{sk}$. 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 $\mathtt{sk}$. $\Box$

3.2 A threshold scheme

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

Construction 4 (generic, threshold, exponentially large domain): *Let a cryptographic scheme with unique secret keys with a relation $\mathtt{Pred}$ as above be given. Additionally, we need a secure commitment scheme that is unconditionally binding; e.g., as in [25].*

- *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 $\alpha_1, \ldots, \alpha_{2^n-1}$ 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 $\mathtt{aux}$ so that $(\mathtt{sk}, \mathtt{aux}, \mathtt{pk}) \in \mathtt{Pred}$. He first proves to the server in zero-knowledge that $\mathtt{pk}$ 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, $\mathtt{pol}(x) \stackrel{\mathrm{def}}{=} \sum_{j=0}^{\tau-1} c_j \cdot x^j$ with $c_0 = \mathtt{sk}$. This corresponds to setting up a secret sharing scheme as in [31]. 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* pol *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* pol *so that $(\mathtt{sk}, \mathtt{pk}) \in \mathtt{Pred}_2$, the non-constant coefficients of* pol *correspond to the commitment C, the constant term c_0 equals* sk*, and $\sigma_\ell = \mathtt{pol}(\alpha_\ell)$.*
- *Verification: The user gives a zero-knowledge proof that he knows a correct pair (σ_ℓ, π_ℓ).*

Proposition 4 *Construction 4 satisfies*

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_ℓ. These are $\tau - 1$ shares of a secret sharing scheme with threshold τ and therefore give no information about the secret sk [31]. □

4 Discrete-Logarithm Schemes

We now present a practical self-delegation scheme for cryptographic schemes based on discrete logarithms in groups of prime order. Important examples are Schnorr and DSS signatures [30, 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 $\mathtt{sk} \in \mathrm{GF}(q) = \{0, ..., q-1\}$, and the corresponding main part of the public key is $\mathtt{pk} = g^{\mathtt{sk}} \bmod p$.

The actual scheme is a slight extension to Feldman's Verifiable Secret Sharing (VSS) scheme based on discrete logarithms [17]. The validation tags are simply shares of the secret key. We need two extra properties for our application: First, if we want a large domain of limitations indices, the VSS scheme must allow a large number of potential shares with an efficient way to index them. Secondly, in our case it is not the shareholders who need to verify the validity of the shares (the shareholders in the motivating example are the laptops), but external verifiers, i.e., we need an efficient zero-knowledge proof of knowledge of a proper share. Although these extra properties are not required in the general definition of VSS (cf. [12]), they happen to hold in Feldman's VSS. This gives the following scheme, for any $\tau \geq 2$.

Construction 5 (discrete log, threshold, exponentially large domain): *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. The domain of limitations indices is $\{1, ..., q-1\}$. The server also selects an arbitrary threshold τ.*
- *Registration: Given a primary key pair $(\mathtt{sk}, \mathtt{pk})$ from the underlying discrete logarithm scheme, the user randomly selects a polynomial $\mathtt{pol}$ of degree $\tau - 1$ over $\mathrm{GF}(q)$ with $\mathtt{sk}$ as its constant coefficient. That is, it selects uniformly coefficients $(c_1, ..., c_{\tau-1})$ in $\mathrm{GF}(q)$. This polynomial is the extended primary secret key. The extended primary public key consists of $\mathtt{pk}$ and the values $h_j = g^{c_j} \bmod p$, for $j = 1, ..., \tau - 1$. The server verifies that these values lie in the group generated by g (by verifying that their q-th power is 1).*
- *Secondary key validation: The validation tag for this user's secondary key for the limitations index ℓ is the value $\mathtt{val}_\ell = \mathtt{pol}(\ell)$, i.e.,*

$$\mathtt{val}_\ell = \mathtt{sk} + \sum_{j=1}^{\tau-1} c_j \ell^j \bmod q.$$

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

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

  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 $\mathtt{val}_\ell$, i.e., of the discrete logarithm of $\mathtt{pk}^*_\ell$ (cf. [9, 19]).[4] An alternative is Schnorr identification [30], which needs only one exponentiation on each side, but is only known to be witness-hiding [18].

Proposition 5 *Construction 5 satisfies*

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 safe choice of the parameters (q, p, g).*
2. Controlled propagation: *Any τ valid secondary key triples yield the primary secret key.*

Proof sketch: The first part follows immediately from the perfect security of the VSS scheme when the public key is already given [28], and the fact that in addition the user only gives zero-knowledge proofs. For Part 2, the server's verification in registration guarantees that the user sets up valid verification information

[4] The system from [9] 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. [19]. However, t need not be very large, because we do not try to exclude every single case of propagation anyway.

for a polynomial over GF(q). The proofs of knowledge guarantee that a person having a valid secondary key triple can extract a value that passes verification as a share. Now the VSS scheme guarantees that τ such shares enable reconstruction of the uniquely defined secret. □

If the set of limitations indices is much smaller than q (e.g., if it corresponds to days of a year), efficiency is greatly improved since the evaluation of $\mathtt{pk}_\ell^*$ by a Horner-like schema only involves exponentiations with small numbers. That is, $\mathtt{pk}_\ell^*$ can be computed from the h_j's and ℓ by iteratively computing $v_{i+1} = (h_{\tau-i} \cdot v_i)^\ell \bmod p$, starting with $v_1 = 1$ and ending with $\mathtt{pk}_\ell^* = v_\tau \cdot \mathtt{pk}$.

An alternative way of using this core scheme is to use the values $\mathtt{val}_\ell$ directly as secondary secret keys (i.e., $=$ $\mathtt{sk}_\ell^*$'s), 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 the 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 the Schnorr identification scheme [30], security does hold, as we will now show.

Construction 6 (Schnorr's identification, threshold, exponentially large domain): *Server setup, registration, and the definition of values* $\mathtt{sk}_\ell^*$ *and* $\mathtt{pk}_\ell^*$ *are as in Construction 5. However, identification with respect to limitations index* ℓ *is immediately conducted with respect to the public value* $\mathtt{pk}_\ell^*$*, i.e., the honest user wishing to identity itself uses* $\mathtt{sk}_\ell^*$ *as the secret key inside Schnorr's identification protocol, and the verifier uses* $\mathtt{pk}_\ell^*$.

Proposition 6 *Construction 6 satisfies*

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

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

A proof sketch is presented in the appendix.

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

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

[9] 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.

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

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

[12] B. Chor, S. Goldwasser, S. Micali and B. Awerbuch: Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults; in *26th FOCS*, pages 383–395, 1985.

[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] 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.

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

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

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] J. Kilian and E. Petrank: An Efficient Noninteractive Zero-Knowledge Proof System for NP with General Assumptions; *Journal of Cryptology*, Vol. 11, No. 1, pp. 1-27, 1998.

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

[26] 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.

[27] T. Okamoto and K. Ohta: Disposable Zero-Knowledge Authentications and their Applications to Untraceable Electronic Cash; in *Crypto '89*, LNCS 435, Springer-Verlag, Berlin 1990, pages 481-496.

[28] T. P. Pedersen: Distributed Provers with Applications to Undeniable Signatures; in *Eurocrypt '91*, Springer-Verlag, LNCS Vol. 547, pp. 221-242, 1991.

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

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

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

[32] 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: Proof of Proposition 6

Let us first recall Schnorr's identification protocol [30], which is actually a proof of knowledge of a discrete logarithm. Its key generation is exactly as described in Section 4, i.e., it yields public parameters q, p, and g, a secret key $\mathtt{sk} \in \mathbb{Z}_q$ and a main public key $\mathtt{pk} = g^{\mathtt{sk}} \bmod p$. The following protocol is used to prove knowledge of $\mathtt{sk}$:

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 $\beta = x + \alpha \cdot \mathtt{sk} \bmod q$.
4. The verifier accepts if and only if $g^\beta \equiv y \cdot \mathtt{pk}^\alpha \bmod p$.

We rename $\mathtt{sk}_\ell^*$ and $\mathtt{pk}_\ell^*$ from Construction 5 into $\mathtt{sk}_\ell$ and $\mathtt{pk}_\ell$ because they are now real secondary keys.

We show that any restricted (as in Proposition 6) attack, A, against the self-delegation identification scheme can be transformed into an equally successful attack, A', against Schnorr's original scheme.[5] Consider such an attack, A. First it expects to get the extended primary public key ($\mathtt{pk}$, $h_1, ..., h_{\tau-1}$), and then up to $\tau - 1$ secondary secret keys $\mathtt{sk}_{\ell_1}$, ..., $\mathtt{sk}_{\ell_{\tau-1}}$. Moreover, the attacker can ask the user to identify itself

[5] Our argument resembles diverted and blinded proofs which have first appeared in [26, 11] (combined with Pedersen's proof of the Verifiable Secret Sharing).

with respect to many other limitations indices. At the end, the attacker tries to impersonate the user with respect to some limitations index $\ell^* \notin \{\ell_1, ..., \ell_{\tau-1}\}$.

We now describe an attack A' against Schnorr's original scheme. Its input is the user's public key $\mathtt{pk} \in \mathbb{Z}_p^*$.

The attack A' starts by setting the grounds for emulating the attack A. It uses the given pk also as the primary public key in the simulated self-delegation scheme (and thus also the same unknown sk). Now A' uniformly selects $\mathtt{sk}_{\ell_1}, ..., \mathtt{sk}_{\ell_{\tau-1}}$ from $\mathbb{Z}_q$ (so that it can later hand these as secondary secret keys to A), and sets $\mathtt{pk}_{\ell_j} = g^{\mathtt{sk}_{\ell_j}} \bmod p$ for $j = 1, ..., \tau - 1$. The next step is to prepare a suitable extended primary public key. Thus we determine the inverse transformation from the secondary keys to the primary key. Recall that the secondary secret keys should be

$$\mathtt{sk}_{\ell_i} = \mathtt{sk} + \sum_{j=1}^{\tau-1} c_j \ell_i^j \bmod q \ \text{ for } i = 1, ..., \tau - 1.$$

Together with the equation $\mathtt{sk} = \mathtt{sk}$, this is a τ-by-τ Vandermonde matrix. From its inverse, we obtain linear equations of the form

$$c_i = m_{i,0}\mathtt{sk} + \sum_{j=1}^{\tau-1} m_{i,j}\mathtt{sk}_{\ell_j} \ \text{ for } i = 1, ..., \tau - 1.$$

With the same coefficients $m_{i,j}$, we have

$$h_i = \mathtt{pk}^{m_{i,0}} \prod_{j=1}^{\tau-1} (\mathtt{pk}_{\ell_j})^{m_{i,j}} \ \text{ for } i = 1, ..., \tau - 1.$$

Thus, A' sets the h_i's like that, and invokes A providing it with the extended primary public key $(\mathtt{pk}, h_1, ..., h_\tau)$ and the $\tau - 1$ "disclosed" secondary secret keys, $\mathtt{sk}_{\ell_1}$, ..., $\mathtt{sk}_{\ell_{\tau-1}}$. Observe that A is invoked on a distribution identical to the one for which its success probability is defined.

Now A will make requests to participate as verifier in self-delegation identification with respect to certain limitations indices ℓ. The algorithm A' handles such a request by requesting an identification in Schnorr's scheme, playing the role of the verifier. It thus obtains from the user some $y = g^x$, which it forwards to A. When A responds with a challenge α, algorithm A' sets its challenge to

$$\alpha' = \alpha \cdot (1 + \sum_{i=1}^{\tau-1} m_{i,0}\ell^i) \bmod q.$$

When the user responds with β', algorithm A' responds to A with

$$\beta = \beta' + \alpha \cdot \sum_{i=1}^{\tau-1}\sum_{j=1}^{\tau-1} m_{i,j}\mathtt{sk}_{\ell_j}\ell^i \bmod q.$$

We show that this response is correct, i.e., that indeed $\beta \equiv x + \alpha \cdot \mathtt{sk}_\ell$ (everything modulo q). (Note that $\mathtt{sk}_\ell$ and also the c_i's are uniquely defined by the extended primary public key, although unknown to the algorithms.) We know that the user's answer is $\beta' = x + \alpha' \cdot \mathtt{sk}$. Thus

$$\beta \ \equiv \ (x + \alpha' \cdot \mathtt{sk}) + \alpha \cdot \sum_{i=1}^{\tau-1}\sum_{j=1}^{\tau-1} m_{i,j}\mathtt{sk}_{\ell_j}\ell^i$$

$$
\begin{aligned}
&\equiv x + \alpha \cdot (1 + \sum_{i=1}^{\tau-1} m_{i,0}\ell^i) \cdot \mathtt{sk} + \alpha \cdot \sum_{i=1}^{\tau-1}\sum_{j=1}^{\tau-1} m_{i,j}\mathtt{sk}_{\ell_j}\ell^i \\
&\equiv x + \alpha \cdot (\mathtt{sk} + \sum_{i=1}^{\tau-1}(m_{i,0}\mathtt{sk} + \sum_{j=1}^{\tau-1} m_{i,j}\mathtt{sk}_{\ell_j})\ell^i) \\
&\equiv x + \alpha \cdot (\mathtt{sk} + \sum_{i=1}^{\tau-1} c_i\ell^i) \\
&\equiv x + \alpha \cdot \mathtt{sk}.
\end{aligned}
$$

When A finally announces that it wants to try to impersonate with respect to some limitations index $\ell^* \notin \{\ell_1, ..., \ell_{\tau-1}\}$, then A' announces an attempt to impersonate in the basic Schnorr scheme. Algorithm A' forwards A's first message, some $y \in \mathbb{Z}_p^*$, to the verifier it wishes to fool. When the verifier responds with a challenge α', algorithm A' forwards $\alpha = \alpha'/(1 + \sum_{i=1}^{\tau-1} m_{i,0}\ell^{*i}) \bmod q$ to A. The divisor is not zero: It is the coefficient at $\mathtt{sk}$ when $\mathtt{sk}_{\ell^*}$ is represented as a linear combination of $\mathtt{sk}$ and the $\mathtt{sk}_{\ell_i}$'s using the inverse Vandermonde matrix. (If it were zero, $\mathtt{sk}_{\ell^*} = \mathtt{pol}(\ell^*)$ with $\ell^* \notin \{\ell_1, ..., \ell_{\tau-1}\}$ would be a linear combination of the values $\mathtt{sk}_{\ell_i} = \mathtt{pol}(\ell_i)$ with coefficients only dependent on the ℓ's. This would contradict the security of the underlying secret sharing scheme.) When A now responds with a value β, algorithm A' responds to the verifier with $\beta' = \beta - \alpha \cdot \sum_{i=1}^{\tau-1}\sum_{j=1}^{\tau-1} m_{i,j}\mathtt{sk}_{\ell_j}\ell^{*i} \bmod q$. We show that if β was a correct response, i.e., $g^\beta \equiv y \cdot \mathtt{pk}_{\ell^*}^\alpha \bmod p$, then β' is also correct: First, we see that y is of the form g^x with $x \equiv \beta - \alpha \cdot \mathtt{sk}_{\ell^*}$ (everything modulo q). Thus

$$
\begin{aligned}
\beta' &\equiv (x + \alpha \cdot \mathtt{sk}_{\ell^*}) - \alpha \cdot \sum_{i=1}^{\tau-1}\sum_{j=1}^{\tau-1} m_{i,j}\mathtt{sk}_{\ell_j}\ell^{*i} \\
&\equiv x + \alpha \cdot (\mathtt{sk} + \sum_{i=1}^{\tau-1} c_i\ell^{*i}) - \alpha \cdot \sum_{i=1}^{\tau-1}(c_i - m_{i,0}\mathtt{sk}) \cdot \ell^{*i} \\
&\equiv x + \alpha \cdot (1 + \sum_{i=1}^{\tau-1} m_{i,0}\ell^{*i}) \cdot \mathtt{sk} \\
&\equiv x + \alpha' \cdot \mathtt{sk}.
\end{aligned}
$$

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.