Split-and-Delegate: Threshold Cryptography for the Masses

Daniel E. Geer, Jr.[1] and Moti Yung[2]

[1] @stake Inc. geer@atstake.com.
[2] CertCo Inc. moti@cs.columbia.edu.

Abstract. Threshold Cryptography (distributed key splitting) is traditionally employed as means to preserve the whole key against compromise, i.e., for risk reduction (coping with memory compromise) and availability (coping with denial of service). Recently, some functionality of splitting keys has been shown to be useful beyond preservation, yielding a small number of high-security, server-related applications. However, the business applications and market applicability of splitting keys is still not realized or analyzed. The goal of this *position paper* is to put forth the thesis that the full power of threshold cryptography as a useful and attractive tool is going to be unleashed only if the ability to split keys is given to end users (the masses). More specifically, we claim that threshold cryptographic operations (e.g. splitting a key) together with user capability to delegate (which we view as a necessary extension of PKI) should be part of the suite of operations available to end-users of a PKI (e.g., embedded in user crypto-APIs / user smartcards). This new tool ("split and delegate") will enable flexible key management at the user level, in contrast with the traditional rigidity of PKI. We note that threshold cryptography is currently mainly an idea and still not in the market (though some companies do offer split key in hardware or software). We believe that the economic value of the suggested user-based applications will be the central driving force behind any market adoption of threshold cryptography. We give an analysis of the potential business and of market penetration scenarios (such business analysis of suggested new cryptographic applications is often done but rarely published).

Key words: PKI, applications, enabling technology, user Crypto-API, smartcard, threshold cryptography, splitting keys, delegation, business analysis, market penetration.

1 Introduction

All available security technology manages the relation between some class of secrets and some class of truth. It is the mapping between secrets and truth that determines all other characteristics of the particular regime. Public Key Infrastructure (PKI) is the generic catch-all phrase for the apparatus of security by way of public key technology. This public key technology creates an asymmetry of keys: a private key half and its corresponding public key half. PKI, in

fact, is about managing the promiscuous distribution of the public half of keys and the chaste protection of the private halves. Virtually all PKI's costs are incurred during verification of the validity (un-revoked status) of a key already in circulation.

Threshold cryptography takes this notion of key halves further and splits the private key-half into fragments and distributes those fragments amongst different holders. Its first refinement, quorumed fragmentation, mitigates the costs of high availability for the fragments as only m-of-n fragments are collectively necessary to recreate the original key. One's cost-benefit-risk tradeoff is between complexified key (fragment) management and the complexified barriers a defrauder must overcome. Putting it differently, quorumed key fragmentation is at once the most powerful current design weapon against both single actor fraud and single actor denial of service.

Or so the story has been told. As always, security technology is cost-dominated whenever it is construed as a disabling (protective) technology, by which we mean a technology whose principal contribution is that of disabling something (an attack) or someone (an attacker). A disabling technology returns nothing of value, per se. Rather it is a tax on productivity, hard to justify as an ROI (return on investment) proposition, and likely to be resisted unless the tax can be finely subdivided such as in a transaction oriented environment. When a technology can only be described as a disabling technology, even the best example of it will rarely find a market. In the case of quorumed key fragmentation, it has (had) no apparent use except for the most principled certifying authorities (CAs) [6] and servers of similar sensitivity (escrow agency [6], internal CA organization [15], tallying authority [4], proxy signing or encryption server [2, 18]) – in total a seemingly tiny market By contrast, an enthusiastic demand pull requires a focus on enablement, not disablement – a demonstrated reason to buy, a positive ROI that does not depend on avoiding situations you would not want even to simulate. Indeed, key fragmentation has been recently added as a capability of a hardware or a software module by some companies, though still for the purpose of disabling advanced attacks. The question, then, is whether it can be made a centrum for an enabling technology and thus be attractive in the business world.

Cryptographic technology is made available within some systems (based on some initial belief of its applicability) and is not made available with other systems (based on a some initial belief in its inability to contribute). Our goal in investigating threshold cryptography as an enabling technology is to look at the business case of embedding this technology within various APIs and platforms, a focus too often ignored when cryptographic technologies are first implemented. Exactly where threshold cryptography is made available has major implications on its applicability since the availability and durability of code within an architecture, and to which entities it is given, is crucial and lasting as is argued in [20].

We posit that key fragmentation has a wealth of application outside of highly protected services such as root CAs, but this obtains if and only if the fragments can be arranged to not complexify key management but merely extend the applicability of cryptographically sophisticated means. This paradoxical result depends on overturning the conventional formulation of a public key hierarchy such that many costs of a general purpose PKI can be avoided, especially those that are due to the rigors of stranger-to-stranger introduction by third parties and the complexity costs of ensuring that all consumers of credentials are equally informed of revoked identities. This overturning may be essential in any case as, perversely, the procedural complexity of absolute key validation (by way of revocation testing) rises as the square of the distance from the root the subject

key is positioned even though the further the key is from the root the less important its protection is likely to be as a practical matter. This is a complexity cost that runs in precisely the wrong direction and it will defeat economic application of public key cryptography as it is presently defined. PKI components are not now, and we argue cannot become, a proletarian commonplace unless these complexity costs are mitigated.

We begin with an idea on that narrow point. A certificate is a file format in which to bind a verifier to an assertion, that is, the public key half to the so-called "DN" or "Distinguished Name" (along with various administrative details we set aside for the moment). A certificate declares that, at such and such a time, the Issuer committed itself to the binding by jointly signing the DN and the key half (at least) with the Issuer's own (private) key.

The X.509 standard posits a hierarchy of certifying authorities, each issuing key-to-DN bindings and each, in turn, certified by its upstream CA, excepting the "root CA" whose certificate, like Napolean's sovereignty, is self proclaimed. Yet, there is nothing inherently hierarchic about certificate issuance; hierarchy is only an issue in verification, both as to a chain of signatures upward to a root and as a source of evidence that those signatures are made in keys as yet un-revoked. It reflects the hierarchic interplay of power and of risk. A certificate itself can, in principle, be created by anyone; it is only virginal thinking that has lead us to hierarchic PKIs. It is only our lack of imagination that has made personal identity checks relative to some certifying authority the bulk of PKI's design focus and of certificate traffic. In fact, the PGP approach [16] and the SPKI/SDSI [11] approaches to PKI allow everyone, in some sense, to be a CA.

So, relax your presumptions for a few minutes. Put a CA in your wallet. Run it on a smartcard, a PDA, or something hybrid. Generate keys and split them. Issue certificates as you will. Delegate to others what you would have them do for you (i.e., split-and-delegate). Make the everyday use of PK technology one that begins with a personal CA and flows outward by having other authorities accept and endorse what the individual has emitted rather than have remote authorities issue you certificates or, worse, issue both the certificates and the key pairs. Invert the hierarchy, give this power to the masses.

2 Applications of key splitting

"Threshold cryptography" is a highly developed topic, mainly for discrete-log based systems and RSA, (in fact, we cannot and do not attempt to cover the entire body of important cryptographic papers herein). The methodology was started in [7, 3, 12, 8, 9] Provably secure schemes were then designed; see [6, 17]. Then adding adversary capabilities was considered in various works. Proactive systems [22] (and schemes in, e.g. [17, 13]) allow the fragments to be renewed and rerandomized which protects even against a mobile adversary that over time compromises many systems' components. It also allows for the changing of the set holding fragments and resplit of keys [22, 14, 10].

Given the above systems, let us view the technology at a very high level of functionality which is sufficient for application development. The characteristics of any key splitting technology we favor are (1) that a key can be split (n-of-n or k-of-n) and, if needful, dynamically re-split, (2) that fragmentation can quorumed (i.e., only m-of-n fragments are needed, and any m will do), (3) that sign/seal operations can be performed step-wise such that asynchronously and partially completed operations do not leak information, (4) that one fragment of a multiway split can be fixed by design, (5) that a security kernel (cryptographic

device) can refuse to leak key fragments once gotten, and (6) that operations can be assured to be correct (due to robustness) in case parties are not trusted.

For completeness, we give an example (based on known techniques) and recall the splitting of an RSA key into two pieces. An RSA private key is an exponent d which is to be used over a (hashed) message M and produce a signature $S = M^d \bmod N$ (N a composite which is a product of two unknown large primes). Then we can assume that a random element g is given together with its signature: $g^d \bmod N$ (a witness). In this case, the owner of the key O_1 splits its key into two pieces by choosing a random element in $d_1 \in [-N, N]$ and assigning $d_2 = d - d_1$. The owner then sends privately to the second server O_2 the value of d_1 and also publishes in public $g^{d_1} \bmod N$ and $g^{d_2} \bmod N$ (as part of a certificate of the splitting). O_2 can check that the public witness of its share is actually g raised to the share. Also, everyone can check that the two new witnesses correspond to a splitting of the original key by multiplying the two new witnesses and comparing to the old public witness. The above was the split-and-delegate portion of the operation. Then the two parties O_1, O_2 can jointly sign a document. The availability of a witness g, g^{d_i} enables the server which applied d_i to a new message M to prove that indeed the share d_i was used correctly: show that logarithm base g of $g^{d_i} \bmod N$ equals the logarithm base M of $M^{d_i} \bmod N$. This can be done using an efficient zero-knowledge technique, which gives the operation the *robustness* property without compromising the security of this signature scheme.

Note that when we split a key 2-of-2 (n-of-n), each fragment is essential, i.e., if we call the fragments A, B, then a completed signature has embedded in itself proof that both A and B signed or, simply, the satisfaction of the logic "A and B." If we split a key 2-of-3 (based on Shamir's polynomial based sharing, say) and promptly sign with (additional) fragment C (whose partial signature can be added to the final signature), then a completed signature has embedded in itself proof that either A or B signed or, simply, the satisfaction of the logic "A or B." Ipso facto, we have the two functions of a monotone logic, viz., AND and OR. An n-of-n split provides AND, a 2-of-n followed by a single fragment signature provides OR. Because a given fragment may be further split, we can combine AND and OR conditions as required. In other words, the completion of a cryptographic operation like a signature is, in and of itself, proof that the logic embodied in the key splits was satisfied whatever that may have been. That this is largely asynchronous and fully dispersable represents a new construct, one that is the basis for the claim that we have moved from protective and static to synthetic and dynamic uses of the technology of splitting keys. To claim this result as fundamental, it is instructive to look at some preliminary instances of its application.

We note that for some applications, one can designate an explicit multi-signature operation which recognizes individual keys [3] and an explicit delegate signature scheme [21]. The difference between implicit key split and explicit multi-signing was discussed in [15]. In most of our applications the key splitting approach is most suitable and easy to interface, though in some cases a combination of explicit sharing and an implicit one makes sense.

So, why split keys? We show next a few examples. We start from the already recognized applications and move into new areas where end-users can apply the technology by splitting and delegating keys. Note that we do not claim that our applications can only be achieved by split-and-delegate, rather that splitting keys (due to the flexibility of control it enables at the user level) captures in a single mechanism a vast amount of applicable procedures and processes. Given

this fact, the following section suggests market and business analysis based on these applications.

1. **CA:** Loss of a Certifying Authority (CA) key is an expensive failure. Splitting that CA private key half as a quorum and distributing the fragments to independent holders is the strongest risk management known for a CA – no single bad actor then has enough authority to make a fraudulent assertion and no single failed actor is essential enough to derail processing. For a CA of sufficient renown, complexity costs are not germane since nearly any complexity cost will pale in comparison to the (reputation capital) cost of a CA failure subsequent to loss of faith in that private key which is truly the root of the CA's authority. One of the early suggestion of using the technology for a CA appeared in [6].
2. **Revocation:** Just as with a CA, the signature that underscores a revocation assertion must be protected against all attackers and is not really part of an ordinary economic discussion as to means and costs. This is particularly relevant if, as we suppose, a root-key for a revocation hierarchy cannot itself be revoked. (The revocation of a key is always retroactive to a time when the key's protection was known to be good hence a revocation of a signature key repudiates those signatures made since that moment of retroactivity and, further, the assertions so signed likewise lose their validity. Since repudiating a signed revocation arguably reinstates the previously revoked key, this is a confusing path down which none of us will want to go.) Protecting this un-revocable key with via fragmentation seems almost a tautologous requirement.
3. **Escrow:** Keys are escrowed for two orthogonal reasons, surveillance and data recovery. Surveillance requires that the cryptographic system have a hole in it and an acceptable escrow system would require that that hole be very precisely accounted for were it to be utilized. We leave the crafting of auditable holes to politics and its desserts (where [6] observed that threshold cryptography can be used for distributing the power of the surveillance agency). Data recovery is, by contrast, nearly a universal good and a data recovery system would enable the very substantial crime prevention outcome of making cryptographic filesystems the norm rather than the exception. Split your personal private key into a 2-of-3 quorum, retain one fragment for your smartcard in your wallet, put one fragment in your principal computing device (such as your laptop) and put one fragment in the company vault. If any one of these is lost the other two can cope – this covers loss of your smartcard, loss of your laptop, and your employer's interest in your data should you expire. No one fragment is useful to anyone, so your laptop can run a cryptographic file system and you can still lose your smartcard without further risk either of disclosure or data loss and so forth and so on.
4. **Countersignature:** Split a key into a 2-of-2 quorum and half-sign with one fragment. Keep that one fragment on your smartcard and send the other to your countersignatory encrypted in his public key, then send a partially signed message to your countersignatory without worry as to its integrity inasmuch as integrity loss merely results in a signature that does not verify. Substitute n-of-n should 2-of-2 be insufficient. A proxy server signing using this idea has been recently suggested in [2] – the difference is that the user holds only one fragment throughout, so the revocation of this key is under the complete control of the server.
5. **Voting/ Distributed Decision:** Split a key into a majority rules quorum, say 5 of 9 for a nine person Board of Directors, and any vote is complete

when 5 of 9 have signed. Since there is nothing in the resulting signature that reveals whose fragments were involved, this is secret voting per se which we consider a feature. Note that this is "one fragment, one vote," so the fragments can be mapped to shares as easily as heads, etc. If you must have "black-ball" voting, use a separate key for that (or permit individuals to "sign" with a random number thus destroying the ability of the signature to complete). In the latter case, do not enforce a robustness mechanism on individual voters: for a small (constant) number of voting directors, trying all subsets (of, say, 5 of 9) is a doable computational task. We note that using threshold schemes for voting in the context of databases was suggested in [10].

6. **Consensus:** Any n-of-n split is a consensus requirement.
7. **Notarization:** If A and B need to regularly ensure that documents passing between them are notarized, they pick a suitable notary with whom they establish a 3-of-3 quorum. What A issues and B accepts C can notarize as when C performs his operation a checkable signature should result. If C refuses to pass the message forward absent that check, there is no notarization without the electronic swearing of the two parties and no evidence of failed attempts. Conversely, B can refuse to accept a communication from A unless A has had that communication stamped by notary C before delivery to B, something that B can check using only B's fragment. In other words, who is enforcing for whom is tailorable.
8. **Off-line deferral:** Any signature with an m-of-n quorum in which you participate can be signed by m-1, put aside either in a filesystem or on your smartcard, and made valid at the moment of use by completing the m-of-n quorum. Where m=n, then only you can complete it. Where m<n, any holder of an unexercised fragment can complete. The point is using time delay to your advantage. A server application of a similar notion is the magic-ink signature idea where unblinding of a signature is time controlled by a quorum [19].
9. **Authorization:** Except in the particular case where "ownership" of a DN confers the right of citizenship, i.e., where owning a name in and of itself confers beneficial rights, an identity cert conveys too little information to manage anything; authorization information has to be added. If, however, the newly logged-in user exchanges his identity cert for a similarly structured authorization credential (in practical terms, a role-identity certificate), then ordinary clients (like browsers) could handle them transparently whilst the server side could rely on the client bringing his access control list (ACL) entry with him at request time rather than the server having to look up a certified identity on a local subset of some global ACL. This issued-on-demand authorization certificate would have a short lifetime and would be but partially signed, i.e., it would be signed by the issuer in a fragment of a 2-of-2 quorum one half of which was issued to the subscriber as the culmination of subscription (or n-of-n for a larger group of approvers). Its invalid-as-issued status would permit its unprotected yet safe transit through the Internet and would relieve the authorization certificate server from having to perform any authentication of the client. This scalability advantage is substantial and underlies the successive designs of Kerberos, the Open Software Foundation's Distributed Computing Environment (OSF/DCE) and the Kerberos adaptation found in in Windows NT 5.5, et seq. In Kerberos-speak, the presentation of an identity cert is like a ticket-granting request and the subsequent authorization cert is like a new credential encrypted in the session key shared between the client and the ticket-granting service. In

short, the authorization cert can be issued without proof that the client is who he says he is – only if that fact obtains will the client be able to complete the signature and activate the authorization. The implication that completing the signature is an act of liability acceptance should be noted (and might even be encoded in the data covered by the signature thereby establishing the basis for recourse). The split signature, done on the fly, can dynamically control the number and composition of ticket granting approvers without increasing the complexity of the ticket beyond that of a cert.

10. **Expiry bridging:** No credential should be immortal but having overlapping credentials can be a management (synchronization) problem even though overlap is the simplest model for avoiding service interruption at the moment of expiry. Since the holder of a "before" fragment and an "after" fragment can perform his partial signature with both fragments, were he to pass along both results he would ensure that a credential is valid regardless of whether the first partial signature was with a corresponding before or after fragment. In this way, only the first holder of a chain of partial signatures need know which to use or, more interestingly, no one need know the moment of expiry with any precision – this is "start using new fragment" followed by "stop using old fragment" with as much slop in between as is needed. Since laptops are notoriously time de-synchronized, this has immediate application.
11. **Privacy:** I can issue credentials from my smartcard CA that carry no information other than that I issued them. These credentials can bridge between web-of-trust models (like PGP) and hierarchic models (like X.509) inasmuch as I can cooperate with a number of other individuals with whom I share a web of trust. When I issue a credential, anyone who knows my public key out of band can, at least, verify the credential in the usual sense of certificate verification. In other words, since any PK pair is a candidate for certificate issuance, a PK pair in a conventional hierarchy can issue certificates that are part of that hierarchy. However, if a collection of people will issue certificates for each other based on 2-of-n splitting, a MIX network ([5]) is used so that all a surveiller knows is that some member of the group was vouched for by the group without being able to enumerate the group or identify the signers. In this way pseudonymity can be achieved and, for large enough groups, effective privacy as well. Note that this feature is, to some extent, considered a bug by the standardizers (PKIX and X.509) who want certificate chains to have additional extensions saying whether such and such a key has the right to certify others. An extension to this effect appearing in a certificate may, of course, be ignored but it does likely provide a genuine attempt at risk allocation.
12. **Payment:** I share a 2-of-2 fragment with my preferred financial institution, e.g., VISA, and issue a partially signed IOU to the merchant at point of sale. The merchant, having done the same (shared a 2-of-2 fragment with VISA), also partially signs. The merchant sends the order and payment information to VISA where VISA completes both partial signatures. If and only if both signatures complete satisfactorily does the transaction go through. This is much simpler than SET ([23]); each party affiliates with VISA by giving VISA a key fragment doubly encrypted, first in the private key of an accompanying public key certificate and then in VISA's public key. All VISA has to do is (verify and) store these initial submissions. When a sale is made, the customer and the merchant each have to compute one partial signature and VISA has to compute two, the completion of which constitutes a receipt for the actual function VISA provides, namely to buy the transactional risk from the merchant/consumer pair for a consideration. Compare that to the

six certificates, multiple, freshly generated secret keys and over a dozen public key operations that SET requires. SET's degree of statelessness is even maintained. In addition, there can be no mismatch between order and receipt as the matching operation is inherent in VISA's purchase of the transaction.

13. **Corporate procurement:** Same as the above but with a 3-of-3 requirement on the buyer side, i.e., corporate audit/finance partial signature, corporate buyer partial signature and VISA's partial signature. Since the corporation might prefer to have spending limits, it could set key fragments a priori for various spending limits and distribute these as a grant of authority to individuals based on rank or trust. The corporate audit/finance key fragment might even be delegated to some particular corporate buyers so that they had 2-of-the-3 of the 3-of-3 quorum at the outset of the transaction.
14. **Automatic validity checking:** The principle in payments and procurement can be extended. Its crux is that validity checking is replaced by correct collaborative completion according to the intent and therefore style by which the original key was split. Note that if I share my key with many institutes, then I can use a pseudorandom function (applied to the name of the institute) in order to determine the institute share (which determines my share as well in a 2-of-2 split). I can also force more than one entity to participate in this validity checking, where the result of validity is made known to everyone without compromising the distributed separation of power for future validations.
15. **Coupons:** I craft a document and compute some operation on it such as a hash. This hash is fixed as (that is, it becomes) the first fragment of a 2-of-2 quorum. I compute the corresponding second fragment and use that second fragment to partially sign the document. As recipient, you can compute the hash and, ipso facto, you can complete the signature. The resulting document is a validated coupon but your signing it proves that you "read" the document inasmuch as you had to process it to acquire the hash. (Regardless of whether this is 2-of-2, the main idea is that the recipient can compute the remaining fragment but only if he "reads" the document he is signing.) This does not require a certificate and the key fragment used for the upstream partial signing is a throwaway. In other words, we are splitting the same key again and again such that the splits are unique but the verification (public) key is constant. The coupons are inherently use-once inasmuch as part of the text hashed to derive the completing fragment can contain a serial number which need be retained by the redemption center against double spending, and nothing more. (The technical binding of content and keys requires that the splits are random; this implies that a random-oracle hash assumption is needed when a fragment is derived such that many (parallel) random splits are indeed secure (e.g. [13])).
16. **Membership lists:** We split a key much like the above, i.e., multiple 2-of-2 splits on the same key. Each 2-of-2 split is used to pre-sign and distribute a document that, in turn, bears the computable wherewithal to complete the signature. The document is a membership form and the completion of the form and a signature to go with it binds the respondent to membership. The blank form, made unique with a membership number, is distributed to a potential member encrypted in that individual's public key as would be any confidential message to a distant counterparty. When the document is returned to headquarters, the complete signature adds the individual to the membership list. From then on, the member will partially sign (using the key fragment he accumulated during enrollment) and send the partially signed message to headquarters. Headquarters will complete the signature and send

the message on to the other members who can jointly and severally verify the authenticity and appropriateness of the message but need only one key to do this, i.e., the public half corresponding to the self-same private key that has been repeatedly split amongst members. This would, for example, provide an automatable way to diminish spam by forcing a star-patterned routing pattern.

17. **Delegation:** I want to delegate something to my assistant. I split my key into a 2-of-2 quorum, retaining fragment A and sending fragment B to my assistant encrypted in her public key. I send an authorization cert, partially signed in fragment A to my assistant, possibly with a time of applicability in the future. My assistant completes the signature with fragment B turning the otherwise useless proto-certificate into a valid certificate. I can do these fragmentations on a per-use basis or I can make this fragmentation permanent to the relationship with my assistant where she might, say, be authorized to read and reply to my mail but not to sign my name to the final budget document. She could, however, return the budget signed partially in fragment B so that I could, if I was satisfied, complete the signature using fragment A thereby making it possible to have her return data to me on an approval basis but with integrity protection built in inasmuch as the "complete the signature" step would fail should there have been an integrity attack. Since sending the key fragment to my assistant requires a decryption operation on her smartcard, it could be arranged that decrypting a key in that manner had the same "never leaves the card" guarantees that key generation on the card is ordinarily assumed to have. (This is a side point but a useful primitive to insist upon when negotiating firmware upgrades to be installed at time of manufacture.) Delegation to a distributed group of assistants is possible as well. In general, delegation can be combined with other applications of threshold cryptography, e.g., with server controlled off-line deferral to control time, and with distributed proxy encryption [18] to route encrypted data to the assistant with the delegated power.
18. **Virtual corporation:** A group of entities need face-to-face negotiation to consummate a deal of some sort. They meet; their respective smart cards hold their particular private key-halves that permit them to link confidentially (via an IPsec VPN, say) to their home organizations. Another unitary smart card representing the virtual corporation of the moment crafts an n-of-n quorum and distributes the fragments to each participant, first signing it then encrypting it in the public key of each participant so that distribution is secure, non-repudiable and specific. That single card also crafts an unsplit key pair and distributes the public half to each participant. As they work, they encrypt their traffic in that public key forcing all traffic to hub through the holder of the private half. The hub logs (and signs) all traffic using the private key-half of the virtual corporation. Note that all traffic is subjected to the decrypt operation even if the result is unreadable due to super-encryption. Such super-encrypted traffic (confidential messages to headquarters via a VPN, say) is opaque to the other participants but it is still logged against the day it might become necessary for some participant to show that they had said or done thus and so at such and such a moment in time. At the conclusion of the deal, the controlling smartcard is destroyed (and with it the key it held) sealing the log for good. Since the deal will be signed n-of-n, the deal is non-repudiable by each participant as well as demonstratable to a third party adjudicator. If one participant wishes to do so, that participant can also display their traffic as sealed in the log by demonstrating its extraction from that log in a key that that

participant controls. The participants, signing n-of-n, can issue (via the controlling smartcard) certificates that authorize or otherwise enable the deal to be effected on, say, multiple exchanges. When it is time for the corporation to cease existence, the controlling smartcard is destroyed thereby ensuring that the facility of its private key dies and, with it, the virtual corporation. This is a powerful construct.

19. **Elective privacy:** An individual can share a 2-of-2 quorum with any entity who holds private data on that individual (we'll call it a repository). When the individual wishes to grant another third party access to the individual's own private data in the repository, the owner drafts a certificate authorizing the third party access to the individual's data. That certificate can be inherently use-once inasmuch as the data repository front-end must sign it upon presentment by the third party before the back-end would honor it but either the front or back ends could retain it for checking against reuse in a spirit much like the way in which an off-line digital coin is retained by (once presented to) a clearing bank as a check against double spending. The management of digital rights in general is an extension of this thread.
20. **Licensure:** A key necessary to use a resource is split between grantor and grantee. When the grantee wants to use the resource, the grantor's half-signed certificate must be completed by the grantee using, presumably, a key half that was issued to the grantee encrypted in the grantee's public key. Of course, the grantee may have shared his half of the quorum with another, but that would now be traceable at least as far as the grantee. The actual holder of the licensed material, which could well be a third party information warehouse, would accept such certificates only when fully signed, of course. This is preferable to a simple license in that it would require the smartcard of the grantee to be present in order for the resource to be used especially if the grantor were to preset its half of the split key to a fixed computation on the public key of the grantee, i.e., if the 2-of-2 quorum had one share fixed in advance by a computation against the public key corresponding to the grantee's smartcard and returned the other share encrypted in that public key. An extension should smartcards be manufactured with a device key pair burned-in seems obvious.
21. **Trial period / promotion:** An authority can be granted first requiring countersignature, i.e., as a 2-of-2 split, but, after favorable performance, the apprentice might be made a journeyman by providing the other half so that the now ennobled individual had a whole key rather than a mere fragment. This can be extended, of course, such that the promotion could be from "must seek two approvers" to "must seek one other approver" to "can approve on own authority."
22. **Smartcard application loading:** Split a key 2-of-2 and put one fragment and the public half on the smartcard at the time of manufacture. When the holder of the other fragment wishes to put something onto the card, he would partially sign in his fragment and send it to the smartcard. Upon receipt, the smartcard would finish the signature and confirm it with the public key (providing both an authenticity check on the sender and an integrity check on data). This beats a conventional signature inasmuch as another listener not party to public key embedded in the smartcard could not confirm the signature nor actually figure out much about how it was constructed. Substitute "encrypt" for "sign" as appropriate.
23. **Re-keyed cipher without receiver re-key:** In some situations, where the ciphertext stream of a message is the result of encrypting with a rolling member of a 2-of-n quorum, the recipient would have to *complete the en-*

cryption (so to speak) before he could do the decryption, assuming that he had been passed one member of the quorum and a corresponding public key. This beats both embedding a series of data encrypting keys (DEKs) within the message as well as the independent transmission of such a DEK series inasmuch as the recipient would have only one, oddly arranged, decryption process without re-keying, yet the message stream itself would be encrypted with as many keys as the sender desired. (The recipient may be a small device and the rolling of fragments at the sender can be viewed as re-keying which is done by the server only).

24. **Suicide revocation:** A subscriber crafts a suicide note of the form "My key is compromised, revoke your trust in it now," partially signs it with a 2-of-2 split of that very key's private half, and distributes this suicide note widely enough that it is too dispersed to squelch at the moment of need. When that time of need arrives, send the other fragment signed in the certificate's full private key half to all repositories which might have the ability to process the suicide note, including by broadcast (promiscuous) means. Any recipient holding the suicide note can complete the signature proving validity of the need to revoke. Note that this does not require a fixed point of revocation service though one might be handy. Note also that there is no need for a revocation service signature to make the revocation valid though that, too, may be handy. This suicide note becomes the primary evidence rather than the revocation service's signature which mitigates high-availability requirements on the revocation service and also handles the particularly thorny problem of a subscriber whose public key half is embedded in certificates issued by a multiplicity of disparate, non-communicating CAs. Were the subscriber to have sufficient trust in an associate, that associate could be entrusted with the completion (second) fragment so that even the inability of the subscriber to craft the emergency notification could be handled by these same simple means. Finally note that because the suicide note is invalid until its signature is completed, theft of the note does not create a denial of service risk for the subscriber, i.e, no attacker can send the subscriber's suicide note for him.

3 Business Analysis

While market analysis is often done when cryptographic technology is brought to market, it is rarely published. We next analyze the potential market of threshold systems and possible market penetration scenarios, based on the ideas and applications suggested herein.

As best as we can now characterize it, the quorumed splitting of keys creates a primitive boolean logic that is cryptographically enforceable. Specifically, a simple key split is an "AND" operation on the key fragments inasmuch as all of them are necessary and the negation (unavailability) of any one blocks the effect of all the others. In similar vein, quorumed key splits form an "OR" operation inasmuch as when, for an m-of-n split, m-1 have made their contributions any of the remaining n-m are in alternation. Since the presence of an AND and an OR operation generates a boolean logic, it seems fair to characterize what we have here as a fundamental cryptographic logic as in "(generalized) secret sharing scheme" but applied to capabilities. The above is not the only way to generate the logic and for example an AND-OR tree of capability can be generated. Also, capabilities can be expressed by a combination of threshold cryptographic scheme (implicit sharing) and delegation via explicit certificates enabling new capabilities [21].

This key splitting technology seems optimal for multi-hand protocols where it is essential that no intermediate results are valid (or misusable) in and of themselves. They include problems of delegation, accountability, and the binding of arbitrary collections of counterparties into sharply demarcatable virtual corporations. We would argue that the only boundaries that apply in an electronic world are cryptographic ones, i.e., that geography is replaced by cryptography as how one describes what is inside and what is outside some given boundary (of course, we do not dispute that national boarders, language boarders and other such speedbumps can be and will be introduced in cyberspace, but they are quite artificial against the capabilities of information technology whereas cryptography is not!). The above protocols give a much richer vocabulary to "What does this signature actually mean?" This is a key-centric view of the world, rather than a name-centric one.

3.1 Threshold Cryptography Market Penetration Analysis

Let us now discuss the development and business implications of the thesis we are investigating (we note that this is rarely done in cryptographic technology papers, but is quite proper in a paper like ours). The more we ponder the ideas above the more we are convinced of the versatility of key splitting at the user level. We think that key splitting inherently models activities of joint action much more prosaic than the high end uses that motivated their invention. These activities have to be modeled and presented to the user in an intuitive and easy to grasp logic, as part of the user interface. If there is to be any real use of these ideas, a company that would make money from them must be focused on solving small economic problems on the cost side of the customer's ledger rather than being entranced by the generality of the ideas however much that may be compelling. What, then, might be such applications? The question is what combination of our list of applications make a nucleus of a product line – one that can yield an early deliverable and can be built upon for additional products. Here are four/five alternatives; not necessarily optimal, not necessarily complete and not necessarily in order by money-making potential. They are:

(0) Split key cryptographic toolkit
(1) Personal CA and countersignature package
(2) Virtual corporation as facilitator for new trading paradigm
(3) Software licensing for metered use
(4) Data recovery form of key escrow

Let us review these application areas:

(0) Split key cryptographic toolkit A toolkit for split key cryptography in and of itself is (almost) a byproduct of developing ultra-high assurance applications such as root certifying authorities. The target customers might be builders of smartcard operating systems, builders of high assurance systems in general, payment systems integrators and other financial processors, access control systems and government integrators. Thinking like Machiavelli, one would want either to make money off this or, if not, at least make sure that no one else could make any money either (a good quality reference implementation for free, say). If it is to be a free-standing business, the toolkit would have to be a money maker, per se. A team that supplies a strictly for-profit toolkit would have regular conflict between (a) minimizing the separation between the current rev of the product and the state-of-the-art versus (b) maximizing the quality and margin of the current rev. If the principle aim of the endeavor is to get split-key ideas adopted then giving them away gains in attractiveness.

(1) Personal CA and countersignature package. The direct application most likely to be immediately useful, assuming the ability to issue a certificate from a personal (possibly smartcard-based) CA holding a terminal key fragment, is the grouping of countersignature, notarization, authorization and delegation. Being able to control a subordinate's ability to use his superior officer's authority would add value to the role of the subordinate while preventing misuse of the superior officer's authority, as the example of handing a budget for approval up to the titular holder of the signatory key has shown. Delegating the ability to respond to e-mail seems easy to explain to nearly anyone and there are many examples in real life where the formal status of one individual, a licensed architect, say, is used as a cover for many others to do their work only to be signed with the licensed architect's stamp. Delegating power to secretarial staff is another concern in organizations. A first focus, then, would be a personal (assumedly smartcard-resident) function that could unwrap and sequester a key fragment just as surely as several of today's cards can create a key pair within a fundamental guarantee that the freshly created private fragment never leaves the card. Once this is available, the only requirement would be the client functionality (assumedly in the form of browser applets). The rest, i.e., the sign-by-degrees software base, is already in place within a number of high-assurance application code-bases albeit without the finishing touches a mass distribution focus would require. Should the split-key toolkit materialize as a product, sub-licensable access to it would be the only raw material needed. The business would be to sell a reference implementation to smartcard vendors (with the fall back of at least encouraging a consistency of interface via open source methods), to sell applets to end-users (hard to do), and to sell both to groupware suppliers (subject to strongly organized prior intellectual property protection within the very small number of viable groupware firms). It is probable that participation in standards processes would be essential but, as always, the tradeoff is between making a standard and negotiating one.

(2) Virtual corporation as facilitator for new trading paradigm. Looking to that part of the financial world where technology is a weapon and new ways to package risk are how one takes large quantities of money from the market, bridging deals across exchanges via a virtual corporation is surely the most attractive. Or, more precisely, if you expect to permit single deals to bridge public exchanges, you will need the building blocks of a virtual corporation, along with VPNs and hooks into execution systems, to do so. Extending, in effect, the "key-centric" school of PKI thought, creating a key and then splitting it amongst counterparties is precisely the act of creating a virtual corporation. Where such a corporation is single purpose, unrolling a large and complex position while maintaining hedges in multiple markets say, subsequently destroying the private key half destroys the virtual corporation's ability to craft new signatures yet, by not destroying the public half you preserve the ability of anyone to verify those signatures going forward in time particularly if order execution includes a self-signed instance of that public key. A partner in trading systems support would likely be essential to market success inasmuch as the exchange that can leverage its back-end systems to provide clearance, etc., to bridged, virtual corporation transactions is the exchange that will make other exchanges secondary to it. The business here is to build, on behalf of an exchange, the tools they would need to make this possible, i.e, there does not appear to be a consumer play here. Further thought will be required as to the day trader especially around trust and clearance issues. If those could be finessed, there would be no need to involve existing exchanges though the barriers to entry would be overwhelming

should the exchanges decide to embrace the technology while themselves having a piece of the action.

(3) Software licensing for metered use The model of buying software upgrades at periodic intervals and installing them is under some pressure; something a recent article in the Wall Street Journal called "upgrade fatigue." Of course, some percentage of each upgrade is actually necessary to repair past sins (especially those related to security), but the steady appearance of new features finesses the public discussion that would otherwise ensue as to why they have to pay to have a defect repaired. A universal client (the browser), mobile code in the form of applets and a work force that collectively expects location independent computing press hard on the fixed-cost model that corporate IT investing in per-seat software upgrades represents. If software were rented/metered it could be cheaper in the large, but budget officers prefer fixed price purchases over metered purchases because they favor predictability over lowest possible price. If software were universally available but only accessible to the holder of the appropriate license, software management would be much cheaper (no tailoring of the desktop needed) while, at the same time, permitting fixed price or metered use to be independently negotiated beyond the issue of who has legitimate access. Licensing to a known individual holding a known smartcard is probably the cleanest answer and has the useful property that the individual could not give away the license without giving away his personal identity. From the individual's point of view, that makes a networked computer in a strange location a useful device inasmuch as the suite of available software would be exactly equivalent to the suite of license fragments on the smartcard inserted into it. The business here would be a license granting engine that is a key-fragment machine, good hooks to on-line transaction systems, some way to reuse (co-opt) existing license services if any such opportunities exist, and the tools to recast this as a species of delegation such that sub-license could also be factored into this mechanism. Barriers to entry would depend on whether the threshold-crypto toolkit were economically reverse-engineer-able (since a closed system need not tell how it does what it does) and the by-now standard pair of relationships and IP protections. Such a model may receive a boost if liability claims against software flaws begin to stick such that the vendor itself would prefer a model of the customer renting the latest, best version rather than continuing to rely on an out-of-date and insecure one.

(4) Data recovery form of key escrow. Finally, the argument that data recovery can be provided without the side effect of enabling surveillance is quite attractive even if for corporations that issue keys (rather than allowing the user to compute them freshly and in private) there is no way to avoid the surveillance threat entirely. Split-key-based data recovery would enable cryptographic filesystems to be more widely deployed and make them operable by the average idiot ("this plastic card is the key and you just put in that keyhole there, just like a hotel room door"). This option has at least the virtue of lazy rollout and no PKI or revocation problem worth talking about. It also means that a standardization of encryption strength and pervasiveness could be done with a single mechanism, i.e., a multi-function excuse for smartcard rollout. The business would be the key fragmentation gear, the smartcard interface and the integration with a cryptographic filesystem, all under intellectual property protections. Making the corporate escrow box pretty resistant would be essential and that might well be the only sine qua non of selling such a simple idea without fear of immediate preemption by the platform vendors and/or OS vendors. The cryptographic filesystems of today are used by the technologically capable more than they are by the slaves of policy, but we can change that. Ordinary people might be in-

terested in this sort of thing for their own purposes ("When I die, Aunt Agnes and my lawyer will be able to open my will"). The general public already has one working example of split key technology in the form of their safety deposit box and the two-key requirement it embodies.

The level of effort to do any of the above alternatives does not seem daunting assuming that a threshold cryptography toolkit [1] is available and that the smartcards/devices become effectively standardized. To the extent that all legacy applications are coming to have Web front ends, the expression of authorization in the form of role-identity certificates means that the Web infrastructure of a corporation can be used as is. Basic logic which expresses delegation rules, has to be incorporated as a basic intuitive user interface layer. The ability to work with a number of different styles of cryptography is eventually necessary, of course, but the most crucial strategic point is to have something the customer can start with at low commitment cost yet which can be upgraded substantially.

4 Conclusion

Splitting keys is undeniably useful for protecting cryptographic capabilities and increasing their availability. Nevertheless, we now conclude the the main advantage is to decouple from group actions any synchrony constraint yet without producing any intermediate validity (namely atomic distributed operation by a cryptographic service/capability). An incompletely processed split-key signature has no validity and leaks no information about the m or the n. By contrast, full document signatures serially applied (aka multi-signing [3]) create intermediate results that are prima facie valid, which are obviously coordinated and which cannot be made anonymous with respect to what members of the quorum signed. We think that deferring all validity to the atomic moment of signature completion is a very useful building block independent of its anti-fraud application in high assurance systems inasmuch as partial signature leaks no information yet retains integrity protection. We think that to whom an authority is delegated can often be as much a matter of privacy as it is a matter of convenience – a rare juxtaposition. We suspect that the application space of these ideas is much greater than we might now realize. There seems to be a lot here.

References

1. B. Barak, A. Herzberg, D. Naor and E. Shai. The Proactive Security Toolkit and Applications. ACM CCS '99.
2. D. Boneh, X. Ding, G. Tsudik and M. Wong. A Method for Fast Revocation of Public-Key Certificates and Security Capabilities. 10th Usenix '01.
3. C. Boyd. Digital Multisignatures. In H. Baker and F. Piper, eds., *Cryptography and Coding*, Clarendon Press, 1989.
4. R. Cramer, R. Gennaro and B. Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. Eurocrypt '97.
5. D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. CACM Feb '81.
6. A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to Share a Function Securely. STOC '94.
7. Y. Desmedt. Society and Group-Oriented Cryptography: A New Concept. Crypto '87.
8. Y. Desmedt and Y. Frankel. Threshold Cryptosystems. Crypto '89.
9. Y. Desmedt and Y. Frankel. Shared Generation of Authenticators and Signatures. Crypto '91.

10. Y. Desmedt and S. Jajodia. Redistributing Secret Shares to New Access Structures and Its Applications, manuscript.
11. C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas and T. Ylonen. SPKI Certificate Theory. Internet Network Working Group RFC 2693, Sep. '99.
12. Y. Frankel. A Practical Protocol for Large Group-Oriented Networks. Eurocrypt '89.
13. Y. Frankel, P. Gemmell, P. MacKenzie, and M. Yung. Proactive RSA. Crypto '97.
14. Y. Frankel, P. Gemmell, P. MacKenzie, and M. Yung. Optimal-Resilience Proactive Public-Key Cryptography. FOCS '97.
15. Y. Frankel and M. Yung. "Dynamic Fault"-Robust Cryptosystems Meet Organizational Needs for Dynamic Control. Financial Cryptography '99.
16. S. Garfinkel, PGP: Pretty Good Privacy. O'Reilly, 1994.
17. A. Herzberg, M. Jakobsson, S, Jarecki, H. Krawczyk, and M. Yung. Proactive Public Key and Signature Systems. CCS '97.
18. M. Jakobsson. On Quorum Controlled Asymmetric Proxy Re-encryption, PKC '99.
19. M. Jakobsson and M. Yung. Magic-ink signatures, Eurocrypt '97.
20. L. Lessig. Code: And Other Laws of Cyberspace. Basic Books, 2000.
21. M. Mambo, K Usuda and E. Okamoto. Proxy Signatures for Delegated Signing Operation. ACM CCS '96.
22. R. Ostrovsky and M. Yung. How to Withstand Mobile Virus Attacks. PODC '91.
23. SET Secure Electronic Transaction Specification, Book 3 Protocol Definition, v1.0 '97. http://www.setco.org/download/set_bk3.pdf